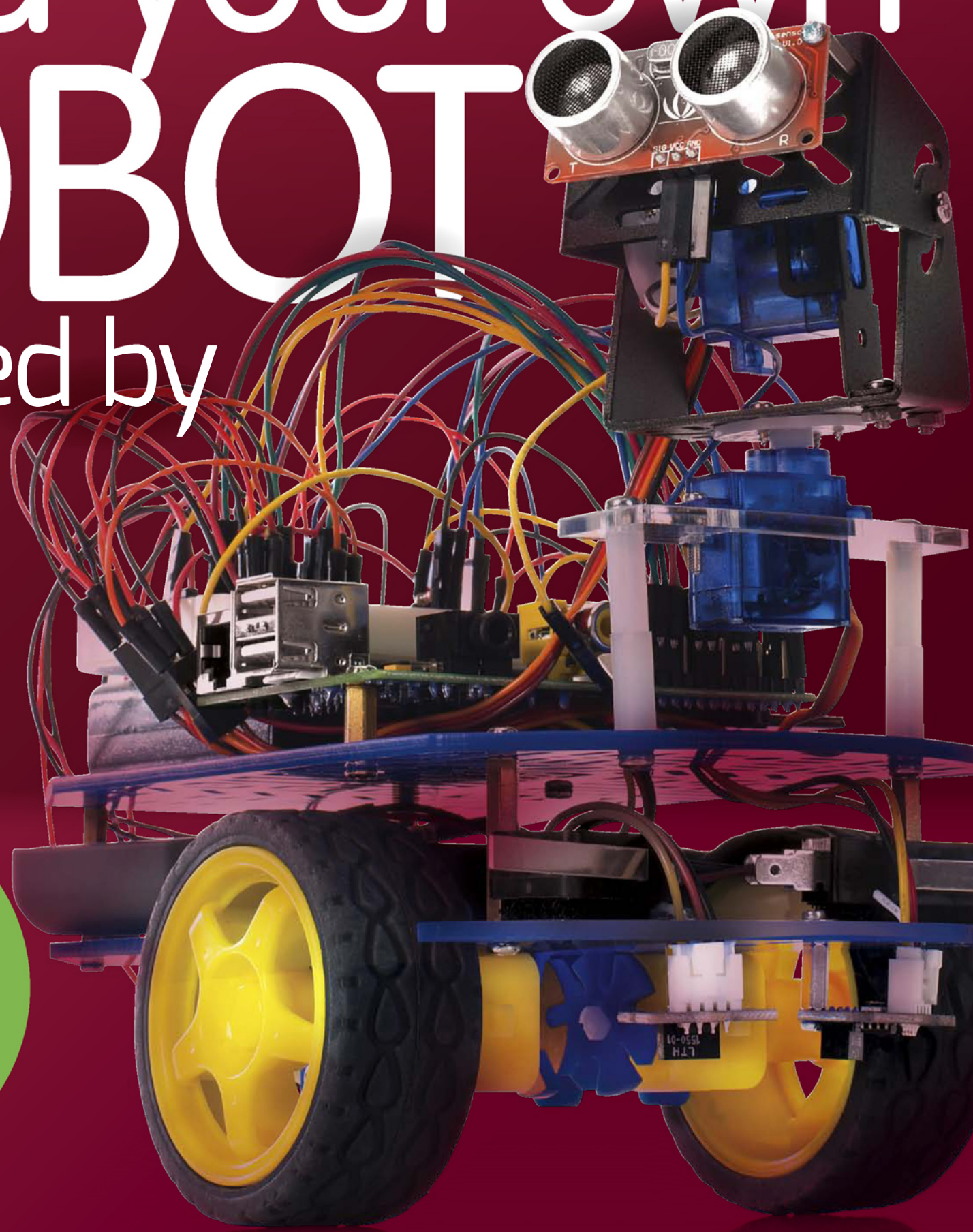
RasPi

DESIGN
BUILD
CODE

2

Get hands-on with your Raspberry Pi

Build your own ROBOT Powered by your Pi



57 PAGES OF RASPI GUIDES

Welcome

Have you ever wanted to build your own self-driving robot? Well, silly question really… So are you ready to build one now? Good! That's exactly what we're going to start doing in this second issue of **RasPi** magazine. We're going to show you all the parts you'll need, how to set up the essentials, build the motor circuit that will power your robot,and then we'll build the chassis. Next time we'll be giving our robot ultrasonic, analogue and line sensors for navigation. And we've got plenty more for you to get your hands dirty with in this issue – you'll be learning to program melodies with Sonic Pi, build a game in Scratch, take photos with your Raspberry Pi and more. Enjoy your issue and have fun!

Gavin Thomas

Deputy Editor

Get inspired

Discover the RasPi community's best projects

Expert advice

Got a question? Get in touch and we'll give you a hand

Easy-to-follow guides

Learn to make and code gadgets with Raspberry Pi

From the makers of

LinuxUser & Developer

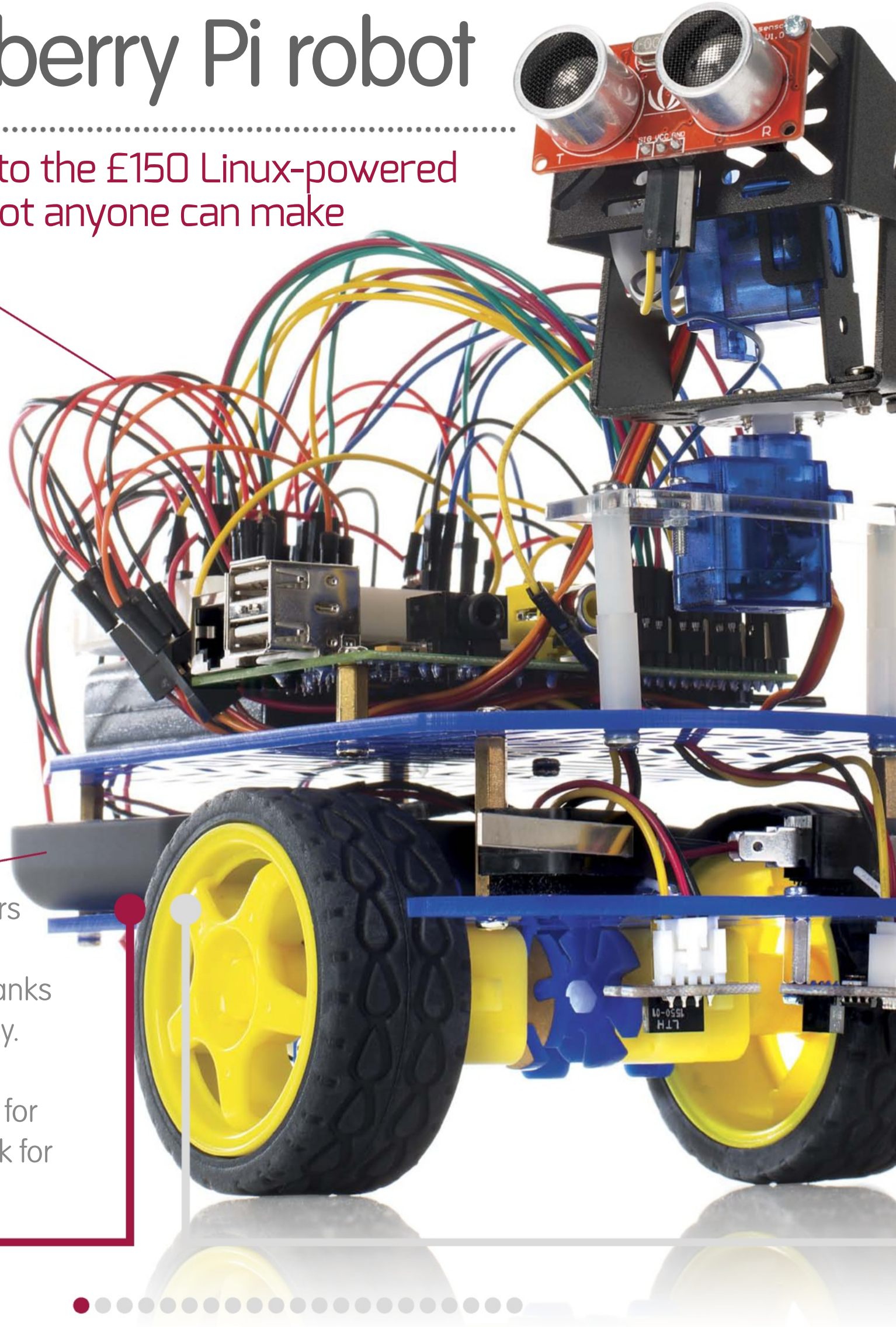
Join the conversation at…

@linuxusermag | Linux User & Developer | RasPi@imagine-publishing.co.uk

Contents

Build a Raspberry Pi robot

Say hello to the £150 Linux-powered robot anyone can make



Spaghetti junction

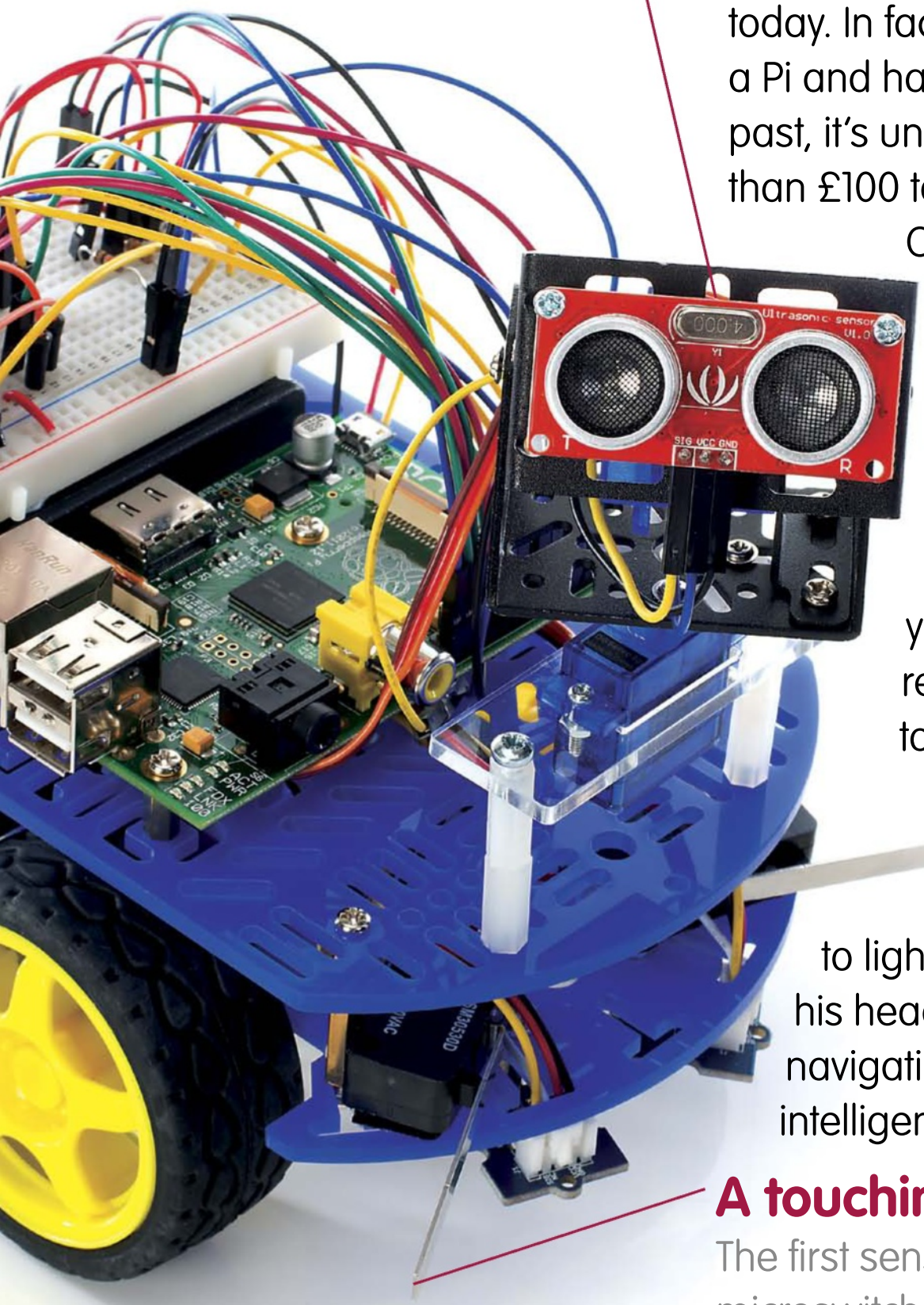
It might look like a terrible tangle of wires now, but by adding motors and sensors gradually and testing and checking as you go, it will soon make perfect sense to you

All aboard

The chassis, motors and wheels are a popular choice thanks to their affordability. As you can see, there's even room for a USB battery pack for the Raspberry Pi

Pleased to see you

While this affordable ultrasonic sensor can't really make our robot see, he will be able to employ echolocation like a bat or dolphin



There's never been a more exciting time to get into robotics. Until more recently even building the most basic robot that moves, senses its environment and reacts to external stimuli cost thousands of pounds construct. Thanks to devices like the Raspberry Pi, though, it can be done at a mere fraction of that price today. In fact, assuming you've already got a Pi and have dabbled in electronics in the past, it's unlikely you'll need to spend more than £100 to put our project robot together.

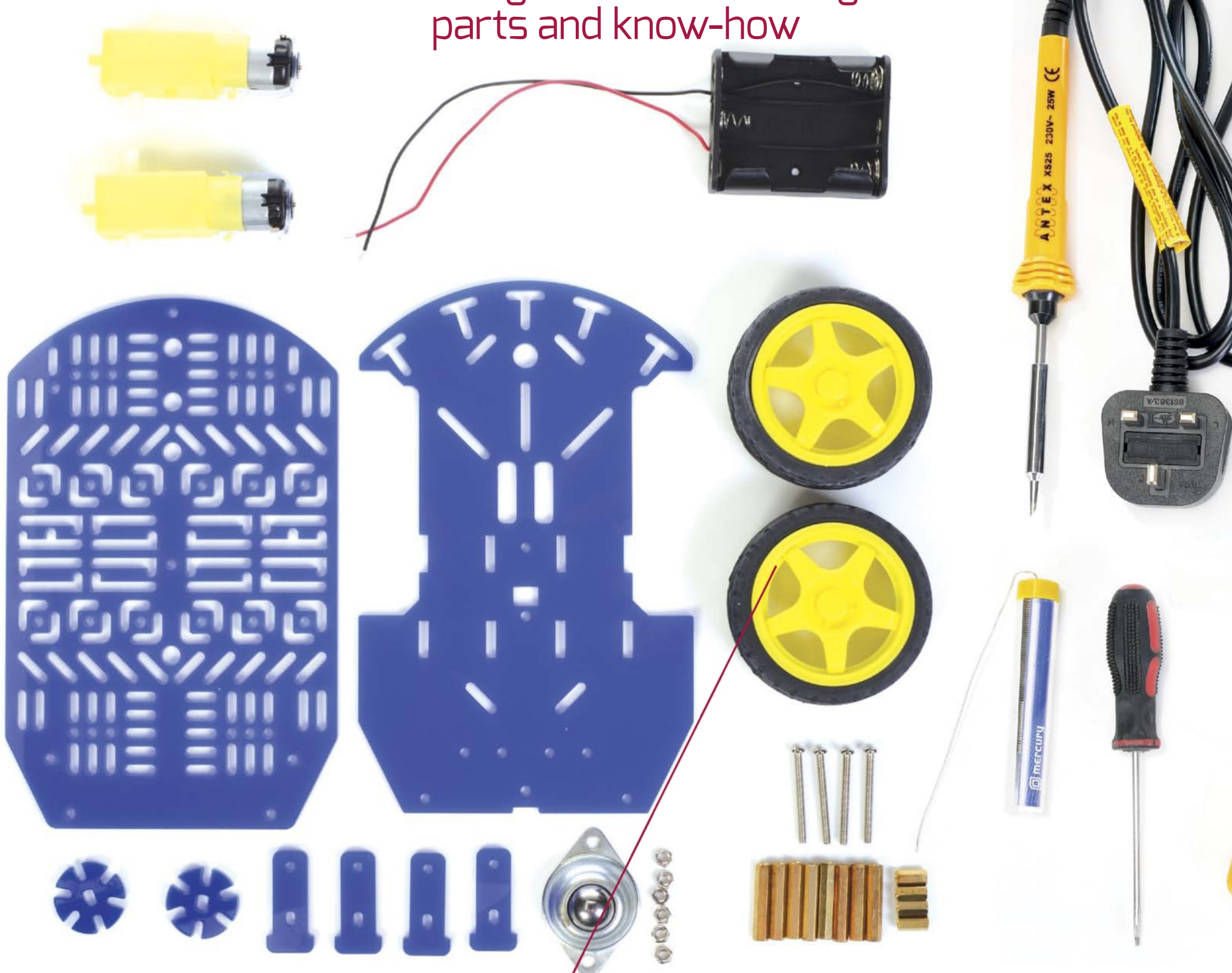
Over the course of the next few pages we'll be exploring aspects of electronics, programming and basic artificial intelligence. You don't need to have any experience in any of these fascinating fields, but we do hope you'll be inspired to learn. We'll be returning to our robot next issue too, giving him microswitches and ultrasonic sensors for collision detection, sensors for tracking lines and reacting to light, plus pan-and-tilt control for his head. The door to exciting fields like navigation, maze solving and artificial intelligence will be firmly open to you.

A touching moment

The first sensors we'll work with are these microswitches or touch sensors. These will enable our robot to react to its environment should it bump into anything

Everything you'll need

Get off on the right foot with the right tools, parts and know-how



Dawn Robotics

www.dawnrobotics.co.uk

Dawn Robotics' Alan Braun knows robots. We relied on his services for this Dagu Magician chassis and the ultrasonic sensor

Pimoroni

shop.pimoroni.com

If you're looking for the best cases, cables and accessories, Pimoroni is an essential stop on your shopping trip. They have a great range of sensors too

Modmypi

www.modmypi.com

We relied heavily on Modmypi's extensive range of hacking and prototyping bits and bobs like breadboards, resistor kits and jumper wires

CPC

cpc.farnell.com

We got our Raspberry Pi, microswitches and some of our tools from CPC. They have a mind-boggling range and the buying power to offer brilliant prices

With our help you'll hopefully find that building a robot with a Raspberry Pi isn't as hard or expensive as you might think. Since there are a number of technical challenges to overcome, you'll need a good selection of electronic prototyping bits and bobs, specialist chips and a few tools to help.

We've laid out many of the core components we've used to make our Raspberry Pi robot in the previous two paages. Don't feel limited to our choices, though. As you'll quickly learn as we make our way through this ambitious project, you can apply the core skills (and even code) needed to access and control the technology to just about any digital or analogue sensors.

Make sure you have a decent small-headed Phillips screwdriver, some decent wire cutters and a soldering iron to hand. While there is very little soldering involved in the initial build, many of the sensors and actuators you'll need later on will depend on them.

If you're looking for the right kit, with the best service at the most competitive prices, you could spend weeks canvassing companies or reading online reviews. Or, you could simply rely on the suppliers we used to put our kit together…

Make and run Python code

You can use whatever development environment you're most comfortable with to write your Python code, be that IDLE, Geany or anything else. That said, there's a lot to be said for simply opening LeafPad, typing some code and saving it as a .py file. It's quicker, more convenient and if you'll learning to code, you'll thank us later.

When it comes to running your scripts or our examples, you need to use superuser privileges or your code can't interact with the GPIO pins.

Caution

While we've carefully constructed this feature with safety in mind, accidents can happen. Imagine Publishing cannot be held responsible for damage caused to Raspberry Pis and associated hardware by following this feature.

This being the case, simply navigate to your file in the terminal and type:

```
sudo python file.py
```

(where 'file' is the name of your code document).

Easy access with SSH

For the ultimate Raspberry Pi robot coding experience we highly recommend kitting out your Pi with a Wi-Fi dongle and SSHing into your Pi from a separate Linux computer. All it requires is that you know the IP address of your Raspberry Pi. You can find it simply by opening a terminal (once you are connected via Wi-Fi) and typing:

```
ifconfig
```

Look for the output that relates to Wi-Fi and make a note of the IP address number. Now open a terminal window on your other computer and type

```
ssh pi@xxx.xxx.x.xxx
```

…using the IP address you wrote down a moment ago. If you've changed the default name from 'pi', don't forget to update that too.

Once you've input your Pi's password (the default is 'raspberry') you'll be connected to your Pi. From here you can navigate to your Python scripts and execute them the usual way. You can even type:

```
nano file.py
```

…to edit your files before running using nano.

Usually there are two options when you're demonstrating electronics – complex circuit diagrams or hard-to-read breadboard photography. Luckily for us there's a third option, which combines the best of both worlds: Fritzing. Fritzing is an open source project designed to support anyone who works with electronics. The tool allows you to pick components and, using a drag-and-drop interface, simply place them on a document and then output it as an image. Hopefully you'll have as much fun using Fritzing with your projects as we did with this one!

www.fritzing.org

Working with the GPIO port

Get to know the GPIO pins on your Raspberry Pi – you won't get far without them

The general-purpose input/output (GPIO) pins on your Raspberry Pi are central to the success of a project such as this. Without them we have no way of interfacing with our motors, sensors or actuators. See the two tutorials later on in this issue to learn more about the GPIO pins.

As you'll soon see, with the help of the Raspberry Pi GPIO Python library it's actually very easy to use them provided you're using the right pin for the right job. Finding the right pin is more challenging that you might think, though, since the pins themselves can actually have several names. For example, GPIO 18 is also pin 12 and PCM_CLK. To save as much confusion as possible, we're using the Broadcom naming convention, as opposed to the board convention. Therefore, in our code you'll see

```
GPIO.setmode(GPIO.BCM)
```

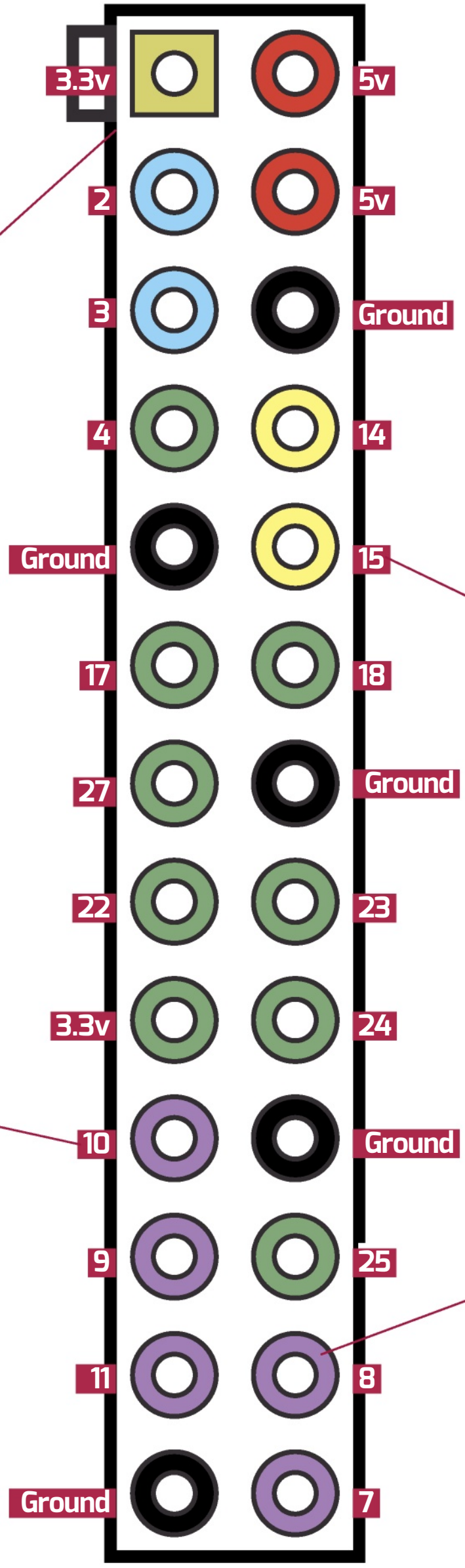
...in all our code listings.

To make matters worse, some pin numbers also changed between Revision 1 and Revision 2 boards. We're using Revision 2 in this diagram (the Raspberry Pi with 512MB of RAM and mounting holes), but you can find the Revision 1 version by searching for 'Raspberry Pi GPIO' online. It can be confusing at first, but the easiest way to deal with the pins is to pick a convention and stick with it!

> "With the Raspberry Pi GPIO Python library it's actually very easy to use the GPIO pins"

This is the top!

The 'top' of the GPIO port here is the end nearest the SD card on your Pi



"To save as much confusion as possible, we're using the Broadcom naming convention"

This is rev 2

There are some different pin numbers depending on your Pi's revision. Don't forget to check!

BCM ,BCM, BCM!

We're using the Broadcom pin numbers, which is a different layout to the 'physical' pin system that can also be used

Purple pins

These pins can be used, but are also reserved for things like serial connections

Build the motor circuit

Let's start by making a simple motor circuit on the Raspberry Pi

Raspberry Pi

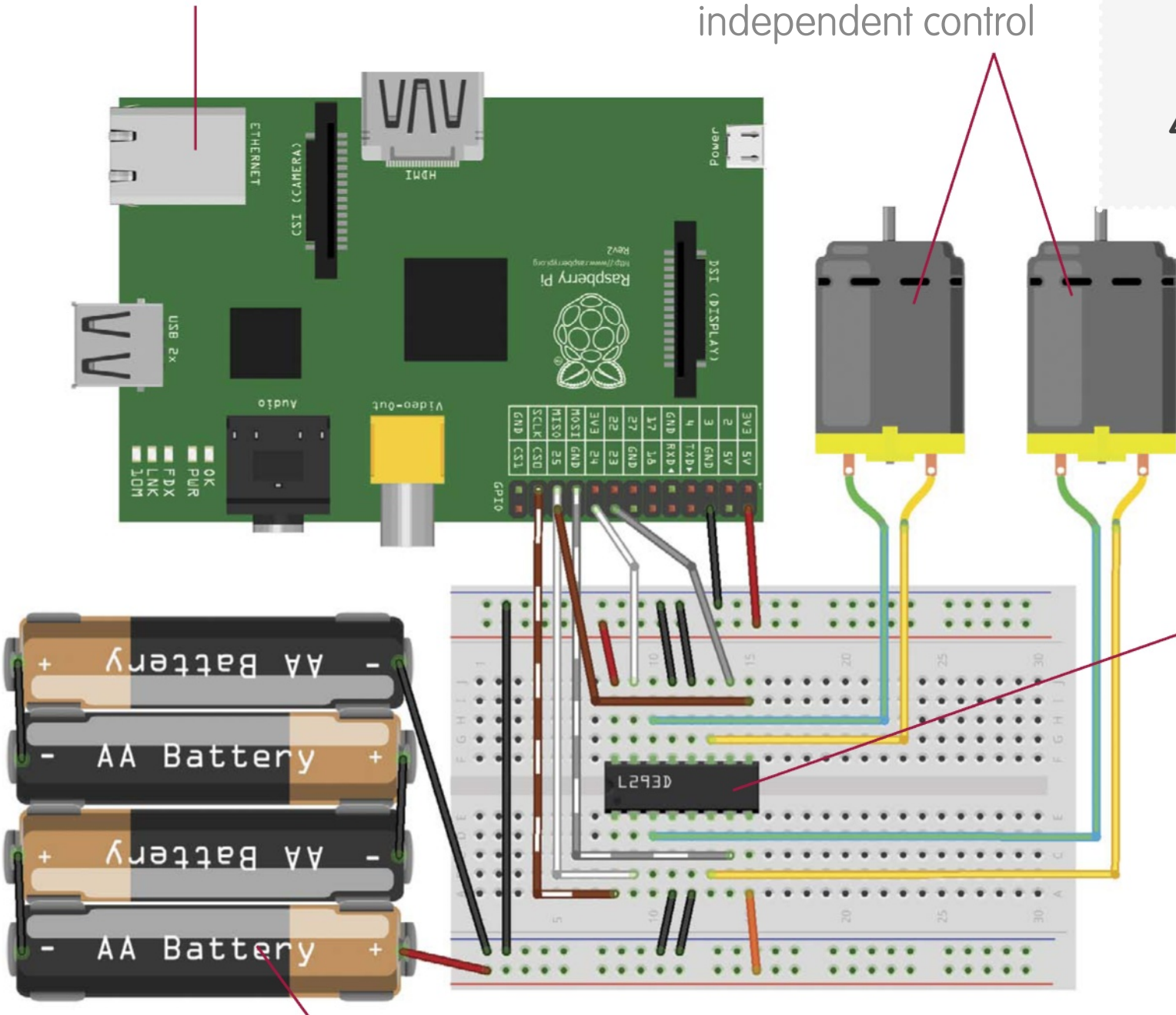
Works with both rev 1 and rev 2 model B, and model A Raspberry Pis

Multiple motors

The single motor driver can handle two separate DC motors, providing independent control

THE PROJECT ESSENTIALS

Raspberry Pi (any model)
Breadboard
2x DC motors
L293D IC
Jumper wires
4x AA batteries
Battery holder



Motor Driver

The L293D sitting across the middle of the breadboard will perform all the hard work

Additional Power

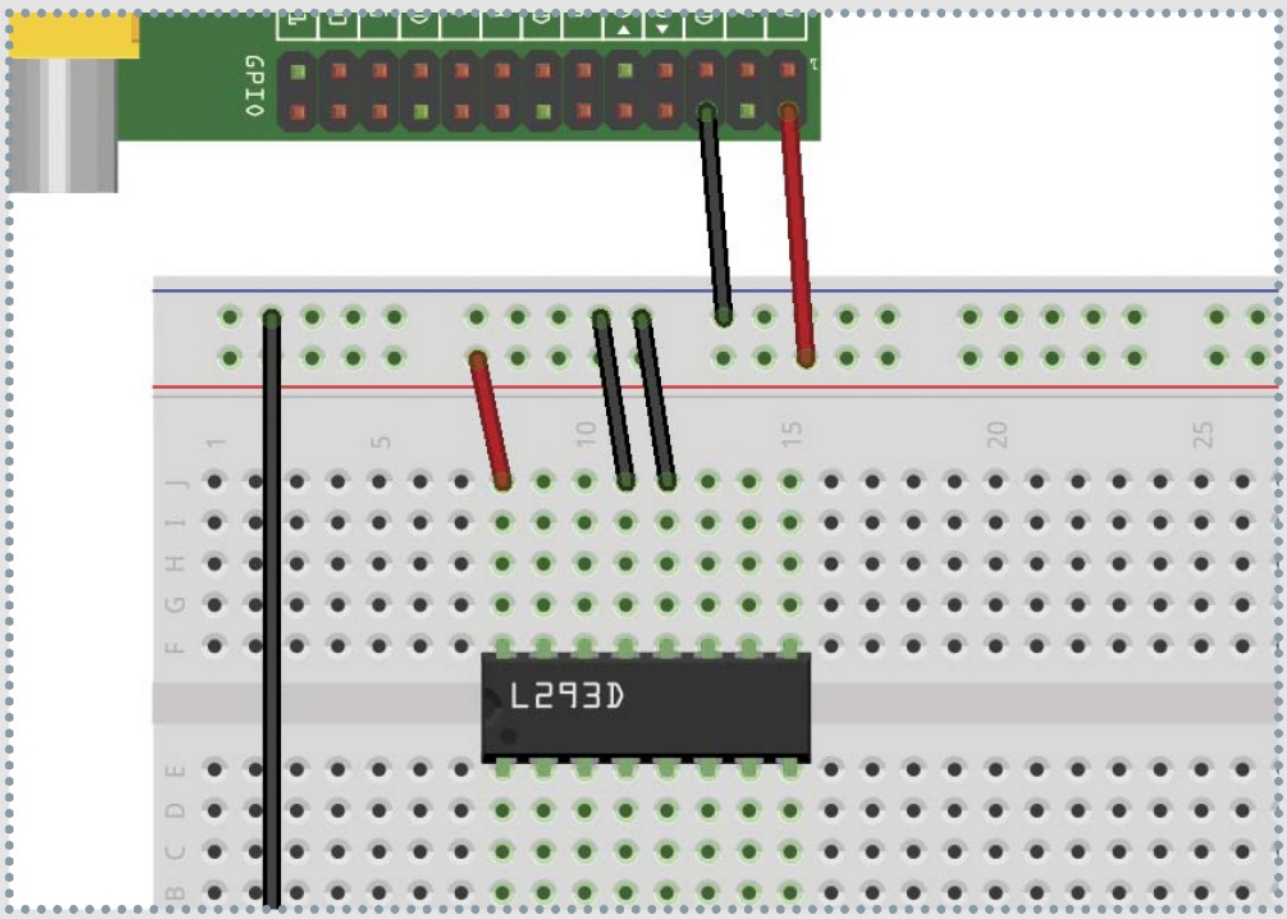
Motors are powered by four AA batteries giving us six volts, perfect for most small robots

The base skill of our robot is movement, and this is handled by the motors supplied with our Dagu Magician chassis. Motors come in a large variety of shapes and sizes, types and models, but here we will safely connect two DC motors to the Raspberry Pi. Due to the limited electrical power offered by the Pi, we will require some additional batteries for a small IC (integrated circuit) to turn our motors on and off for us. Don't ever power them from the Pi.

The motor driver we will use is called an L293D, otherwise known as an H-Bridge. This one IC, or chip as it's sometimes called, will handle the separate power control as well as provide us with bi-directional control for two motors.

01 Adding the L293D

Place the L293D chip into the middle of the breadboard and add the red and black power cables, paying attention to the pins. The orange wire will be for the batteries.



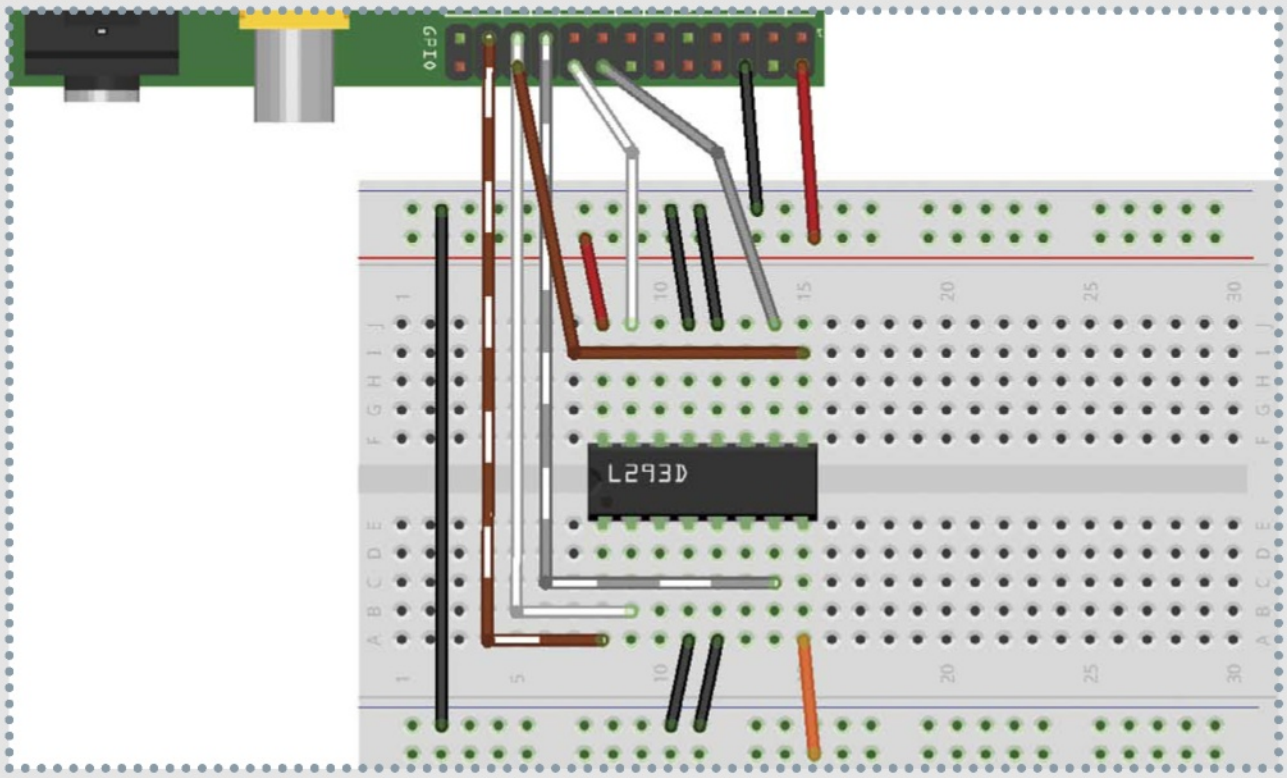
Caution

Never connect motors directly to your Raspberry Pi. Doing so can damage the central processor, resulting in a costly (but attractive) paperweight.

"The motor driver we will use is called an L293D, otherwise known as an H-Bridge"

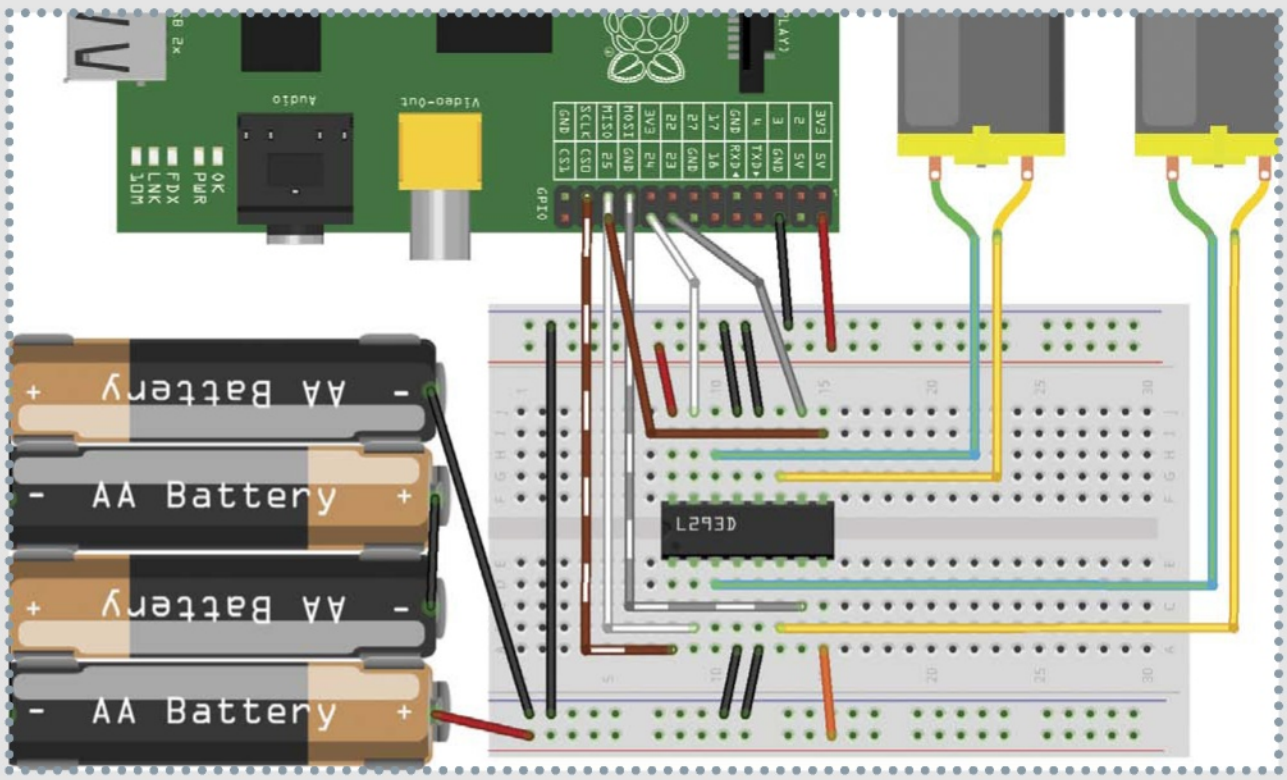
02 Configure the data lines

Double-check the connections to ensure the pins are correct. There are six wires going from the Pi GPIO pins to the input pins of the L293D. These will be responsible for our motors.



03 Finish the circuit

Now we can add the motors. We won't know which way they will turn yet, so make sure you can easily swap them around. Finally, add the batteries and plug in the Raspberry Pi.



"We won't know which way the motors will turn yet, so make sure you can easily swap them around"

First motor test

With your circuit complete, here is how to get your motors moving

Using Python to control the motors is made nice and simple with a library called RPi.GPIO. This gets imported into your script and will handle all the turning on and off that you require. It can also take inputs such as sensors and switches, which we'll cover next time, but first let's make our motors turn to give us some movement.

First we'll import the library we need, RPi.GPIO. We also want to be able to pause the script so we can let the action run, so we'll need to also import the sleep function from the library called time. Next we'll tell the script what numbering we require. The Raspberry Pi has two numbering schemes for the GPIO pins: 'board' corresponds to the physical location of the pins, and 'BCM' is the processors' numbering scheme. In these scripts we'll use the BCM scheme.

It's not necessary, but it's a good idea (to save on confusion later) to give the pins you will use a name. So we shall use the L293D pin names to make controlling them easier. Each motor requires three pins: an A and a B to control the direction, and Enable that will work as an on/off switch. We can also use PWM on the Enable pin to control the motors' speed, which we'll look at after this.

All that leaves us with is to tell the pins they need to be an output, since we are sending our signal from the

> "It's a good idea to give the pins a name. So we'll use the L293D pin names to make controlling them easier"

Raspberry Pi. To turn the pin on – otherwise known as 1, or HIGH – we tell the Raspberry Pi to set that pin HIGH; and likewise, to turn it off, we set the pin LOW. Once we have set the pins, we shall pause the script using time.sleep() to give the motors a few seconds to run before changing their direction.

01 Prepare your script

Login into your Raspberry Pi – username is 'pi' and password is 'raspberry'. Now we'll create our first script, type in:

```
nano motortest.py to begin.
```

This will open the nano text editor.

02 Save your code

Type in the code that's listed across the next two pages, but remember it's case sensitive. Capital letters are important. And indent the code with a space, keeping it consistent. When done, hold Ctrl and press X, then Y to save.

03 Test your motors

Now to run it. For this we type:

```
sudo python motortest.py.
```

If something doesn't work, retrace the wires, making sure they connect to the right pins and that the batteries are fresh.

"If something doesn't work, retrace the wires, making sure they connect to the right pins and that the batteries are fresh."

Below left First we import RPi.GPIO and set the pin naming

Below The nano command is the easiest way to create a new Python file

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)

Motor1A = 24
Motor1B = 23
Motor1E = 25

Motor2A = 9
```

```
pi@192.168.0.18's password:
Linux raspberrypi 3.6.11+ #456 PREEMPT Mon May 20 17:42:15 BST 2013

The programs included with the Debian GNU/Linux system are free sof
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Sep 22 12:00:50 2013 from 192.168.0.14
pi@raspberrypi ~ $ nano motortest.py
pi@raspberrypi ~ $ sudo python motortest.py
```

The Code MOTOR CIRCUIT

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)

Motor1A = 24
Motor1B = 23
Motor1E = 25

Motor2A = 9
Motor2B = 10
Motor2E = 11

GPIO.setup(Motor1A,GPIO.OUT)
GPIO.setup(Motor1B,GPIO.OUT)
GPIO.setup(Motor1E,GPIO.OUT)

GPIO.setup(Motor2A,GPIO.OUT)
GPIO.setup(Motor2B,GPIO.OUT)
GPIO.setup(Motor2E,GPIO.OUT)
```

The start

These are the GPIO pin numbers we're using for our motors. We've named them according to the L293D for clarity

Setting outputs

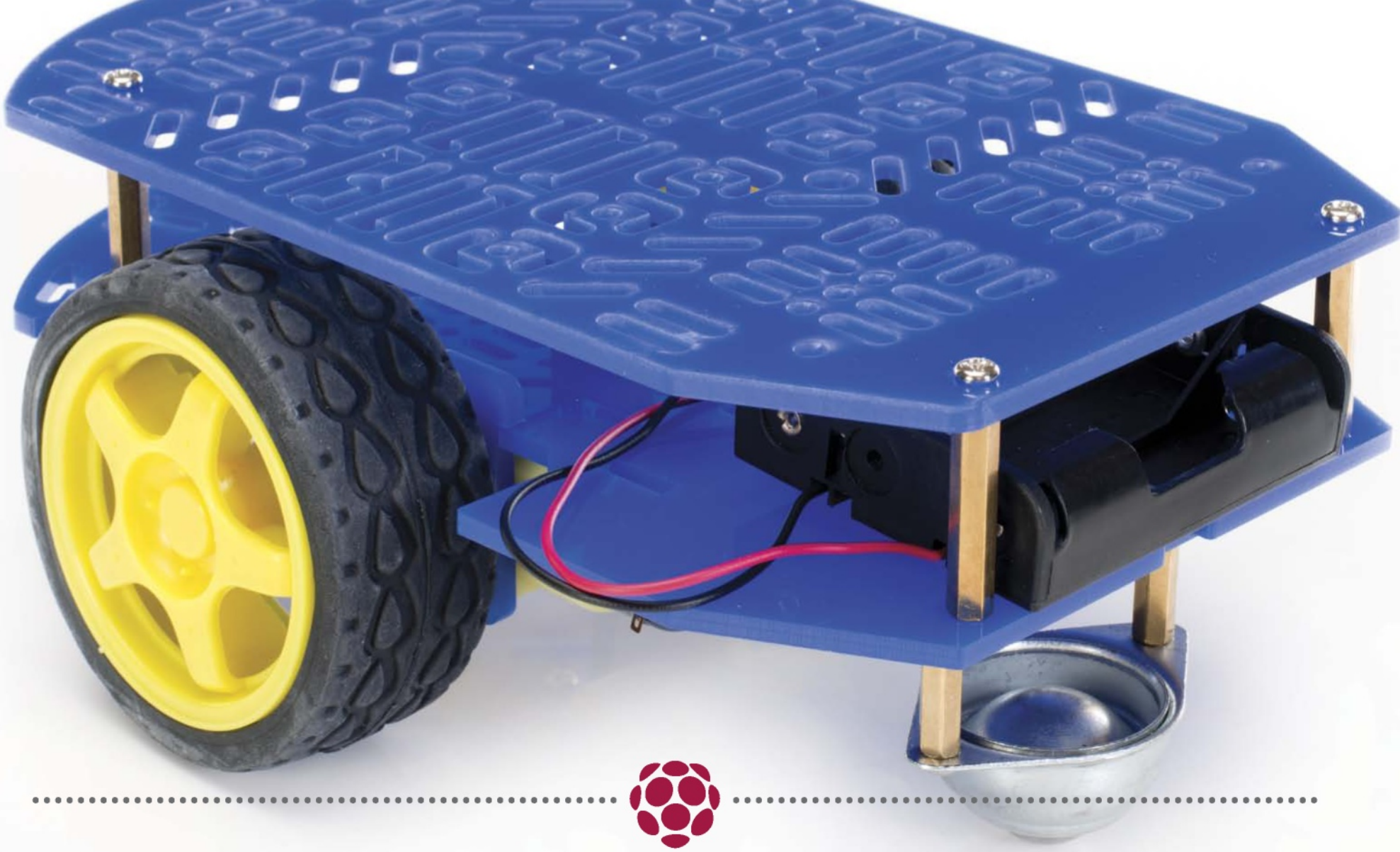
As we want the motors to do something, we need to tell Python it is an output, not an input

The Code MOTOR CIRCUIT

```
print “Going forwards”
GPIO.output(Motor1A,GPIO.HIGH)
GPIO.output(Motor1B,GPIO.LOW)
GPIO.output(Motor1E,GPIO.HIGH)

GPIO.output(Motor2A,GPIO.HIGH)
GPIO.output(Motor2B,GPIO.LOW)
GPIO.output(Motor2E,GPIO.HIGH)

print “... for 2 seconds.”
sleep(2)
print “Going backwards”
GPIO.output(Motor1A,GPIO.LOW)
GPIO.output(Motor1B,GPIO.HIGH)
GPIO.output(Motor1E,GPIO.HIGH)

GPIO.output(Motor2A,GPIO.LOW)
GPIO.output(Motor2B,GPIO.HIGH)
GPIO.output(Motor2E,GPIO.HIGH)

print “... for 2 seconds”
sleep(2)

print “And stop before cleaning up”
GPIO.output(Motor1E,GPIO.LOW)
GPIO.output(Motor2E,GPIO.LOW)

GPIO.cleanup()
```

Making movement

We are now telling the L293D which pins should be on to create movement – forwards, backwards and also stopping



Assemble the robot chassis

Now we've got a working motor circuit, let's start building our Raspberry Pi robot

One thing a robot can't live without is somewhere to mount all the parts, so for this we need a chassis. There are many different sizes, shapes and finishes available. We are going to use one of the most versatile and common, called a Dagu Magician.

This chassis kit comes complete with two mounting plates, two motors and two wheels, as well as a battery box, which is perfect as a starting point for most basic robots. Once this is ready, we can start to expand our robot with new motor functions, before adding sensors.

Building tips

Take your time

It's easy to jump ahead and assume you already understand the build process without glancing at the instructions. That pamphlet is there to help – use it!

1 Sort through the parts

Lay all the parts out and familiarise yourself with them. Assembly is for the most part straightforward; some care is needed with the motor bracket, though.

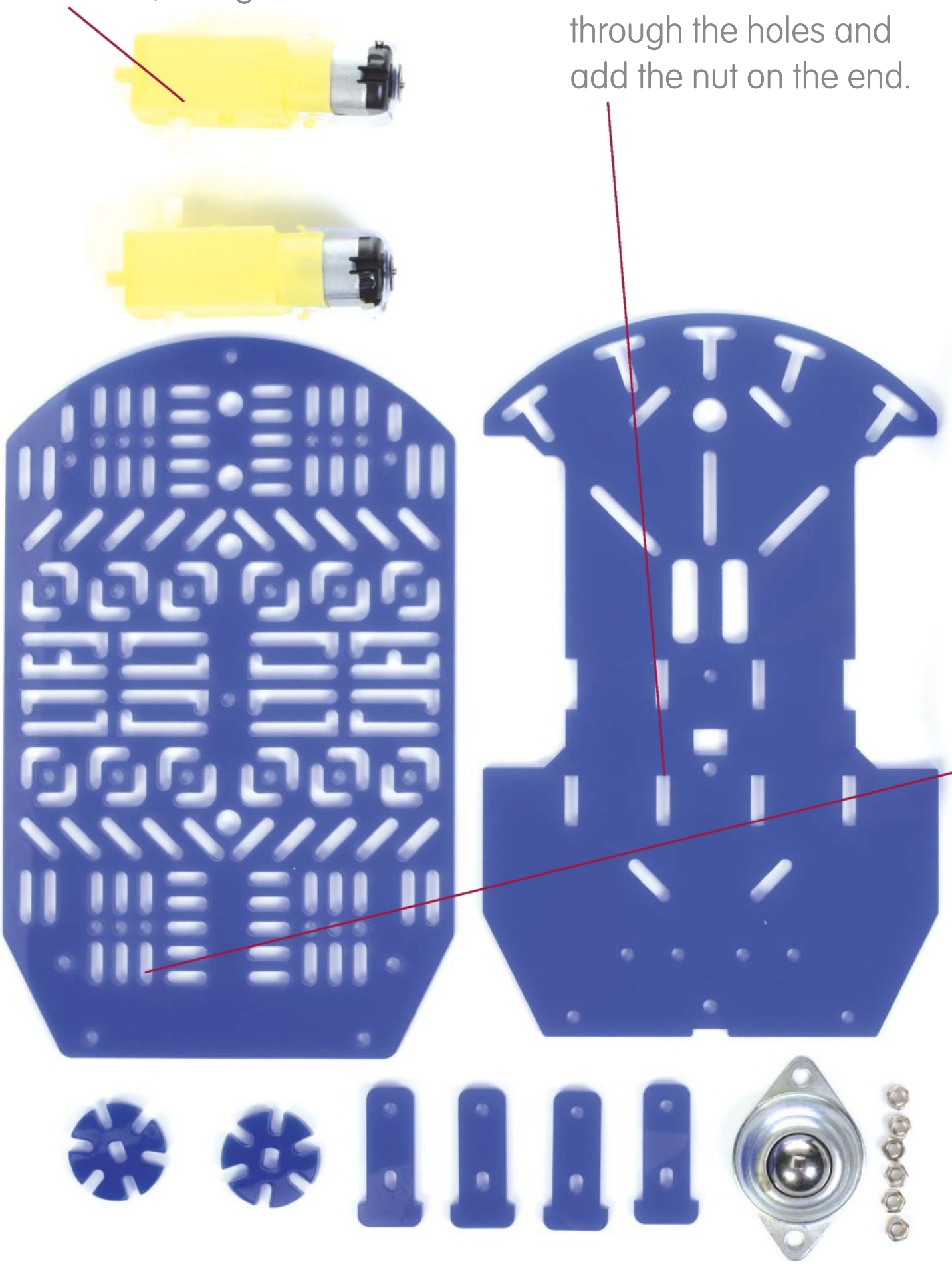
2 Assemble the motor bracket

Insert the bracket through the chassis and sandwich a motor with the second bracket. Feed a bolt through the holes and add the nut on the end.

Building tips

Modifications are welcome

Don't limit yourself to the stock design. If you need to cut a new set of holes for your sensors, measure twice and cut once, but don't feel limited to the stock options.



3 Piece the bits together

Feed the motor wires up through the chassis and add the top plate using the hexagonal spacers and screws, followed by the castor.

04 Wire everything up

With everything in place, using the motor circuit, reconnect the Raspberry Pi again and switch it on. Make sure it works by running the test script.

05 Add the breadboard

Most breadboards have an adhesive pad on the bottom so you can peel and stick down, or use Blu-tack for a less permanent fix. Mount this at the front.

06 Mount your Pi

The Raspberry Pi rev 2 and some cases have mounting holes on the bottom, so utilise them for mounting, and fix the battery packs into place.

"Most breadboards have an adhesive pad on the bottom so you can peel and stick down, or use Blu-tack for a less permanent fix"

Building tips

Plenty of choice

There is a world of choice when it comes to robot platforms. Four wheels, tracks and even hexapods are possible. One of our favourite robot platforms is Frindo.

Create movement functions

Our simple motor test won't do for a finished robot – we need to add more abilities, so let's add some movement functions we can call upon whenever we want

Now that we have a fantastic-looking robot and everything wired in the right place (apart from the motors, which we may have to change), we can plug in the Raspberry Pi and write our first script to make the robot controllable. Our simple motor test from before was perfect for checking if the motors worked and gave us the basics of movement, but we want to be able to control and move it around properly and with precision. To do this we need to create our own functions.

In Python this is done by grouping repetitive actions into a definition or def block. Using the def block we can easily pass parameters such as speed and write the code that controls the pins. We will also add PWM support, so we can set a speed that the motors should run at. In the first few blocks of code, we'll set up the pins we need, setting them as outputs; the next block will tell Python to enable PWM on the two Enable pins.

In the next few blocks we will be starting to create our functions, giving them easy-to-remember names such as forward and backward, but also allowing individual motor controls by using left and right. Up to this point nothing will happen, as we haven't told Python what we want to do with them – we do that at the end. We shall tell the motors to go forward at 100 (which is full power) for 3 seconds, then backwards at full power for 3 seconds.

Repeating code

In Python we use a definition block to repeat sections of code; this allows us to use the same code several times, as well as make changes nice and quickly.

> "We want to be able to control and move it around with precision"

The Code MOVEMENT FUNCTIONS

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)

GPIO.setup(24,GPIO.OUT)
GPIO.setup(23,GPIO.OUT)
GPIO.setup(25,GPIO.OUT)
GPIO.setup(9,GPIO.OUT)
GPIO.setup(10,GPIO.OUT)
GPIO.setup(11,GPIO.OUT)

Motor1 = GPIO.PWM(25, 50)
Motor1.start(0)
Motor2 = GPIO.PWM(11, 50)
Motor2.start(0)
```

```
def forward(speed):
  GPIO.output(24,GPIO.HIGH)
  GPIO.output(23,GPIO.LOW)
  GPIO.output(9,GPIO.HIGH)
  GPIO.output(10,GPIO.LOW)
  Motor1.ChangeDutyCycle(speed)
  Motor2.ChangeDutyCycle(speed)
def backward(speed):
  GPIO.output(24,GPIO.LOW)
  GPIO.output(23,GPIO.HIGH)
  GPIO.output(9,GPIO.LOW)
```

"Save typing which pin needs to do what by grouping them into a definition block"

Set the pins

First we'll import the classes we need, and set up the pins the same as we did for the motor test

Enable PWM support

To allow us to control the speed of the motors, we require pulse-width modulation (PWM). As the Enable pin supports this and works for both directions, we'll set it to this pin

Movement functions

Python allows us to simplify and reuse code, making it shorter and easier to read. We'll use this to save typing which pin needs to do what, by grouping them into a definition block

The Code MOVEMENT FUNCTIONS

```
  GPIO.output(10,GPIO.HIGH)
  Motor1.ChangeDutyCycle(speed)
  Motor2.ChangeDutyCycle(speed)

def left(speed):
  GPIO.output(24,GPIO.HIGH)
  GPIO.output(23,GPIO.LOW)
  Motor1.ChangeDutyCycle(speed)

def right(speed):
  GPIO.output(9,GPIO.HIGH)
  GPIO.output(10,GPIO.LOW)
  Motor2.ChangeDutyCycle(speed)

def stop():
  Motor1.ChangeDutyCycle(0)
  Motor2.ChangeDutyCycle(0)
```

```
forward(100)
sleep(3)
backward(100)
sleep(3)
forward(50)
sleep(5)
stop()
left(75)
sleep(2)
right(75)
sleep(2)
stop()
```

Change speed

With the addition of the (speed) element, we can input a number into the function that it can use and return the result – in our case, the speed of the motor – back into the script

Make it move

Up until now the script will do nothing noticeable, but the hard work is out of the way. To add movement, we shall use our new variables

Fully mobile

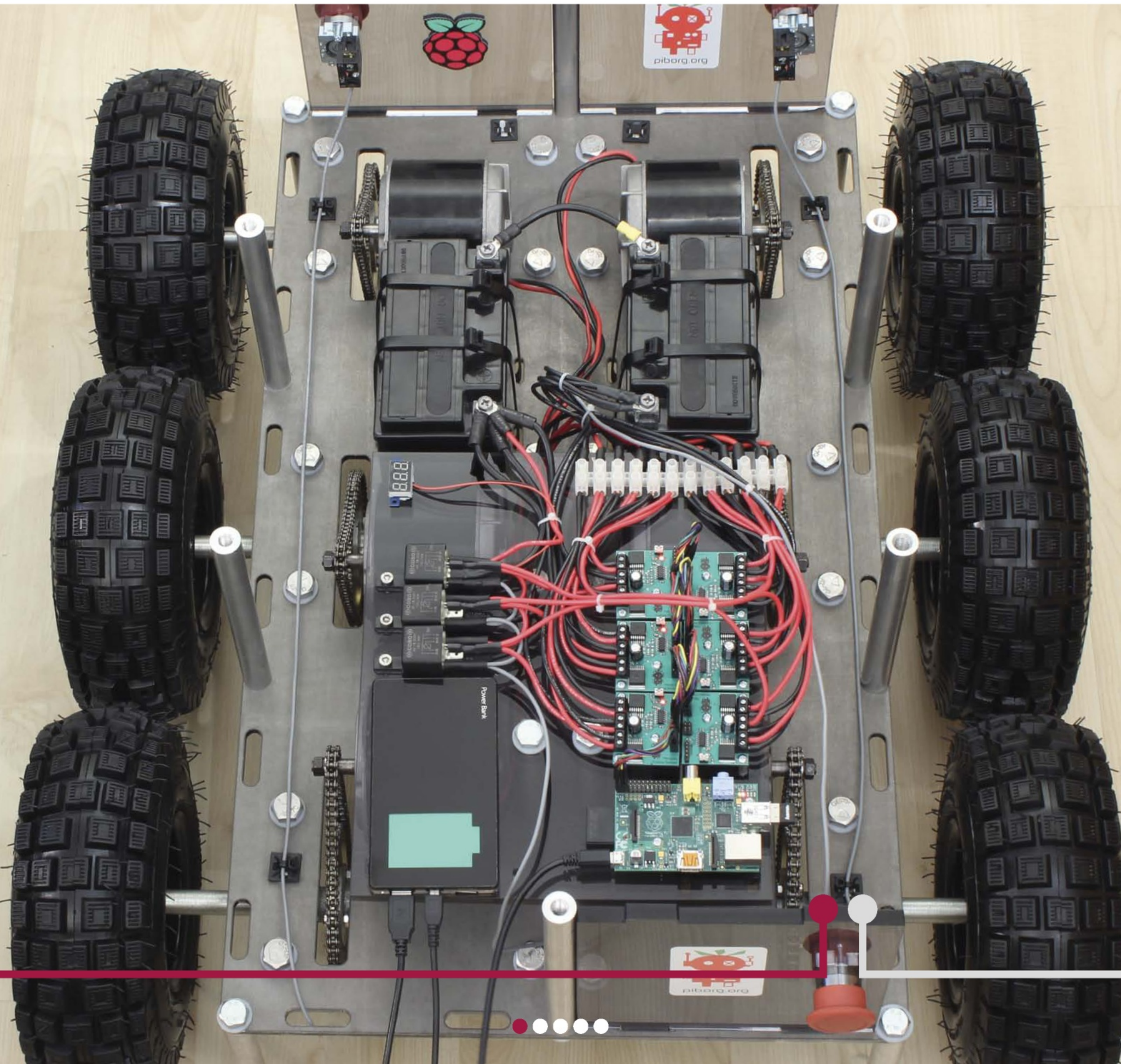
We can also able to control each motor separately with left() and right(), allowing the robot to turn on the spot. Combined with sleep(), it means we have a fully mobile robot!

Pulse-Width Modulation

PWM is a technique used to vary the voltage on parts like LEDs and motors by rapidly switching it on and off.

DoodleBorg

The six-wheel rover robot built by the PiBorg people is a Raspberry Pi-powered beast that can tow a caravan



Can you tell us what's hooked up to a Raspberry Pi that makes it the DoodleBorg?

[DoodleBorg is] running off six motorcycle starter motors. It's using mini all-terrain vehicle wheels and has a custom chassis made out of six-millimetre thick steel that has been laser cut. It has two motorcycle batteries, and six of our wonderful PicoBorg reverse control boards which are capable of five amps per channel, ten amps in total. We've got them connected up, one per motor so we can individually control each of the wheels. This means we can make alternate wheels go in all sorts of directions if we want them to. There are some big crazy switches on the front that serve emergency power-offs too.

"We can pull a trailer, caravan, boat… and maneouvre it into difficult-to-get-to spots"

Tim Freeburn is the director of Freeburn Robotics, the parent company of PiBorg. He has a degree in Mechatronics Engineering

Like this project?

While you won't be able to buy a DoodleBorg kit, there is the mini PiCy robot – a mini Raspberry Pi robot that PiBorg sells

Further reading

To learn more about the DoodleBorg, visit the PiBorg website over at **piborg.com/doodleborg**

How powerful is the DoodleBorg?

It can do 30 miles an hour, although it's not actually geared for speed – it's geared for torque. So what [we have for the DoodleBorg] is a tow ball attachment to go on the top of it. With that what we can do is pull a trailer, caravan, boat or something like that. We can drive it around and manoeuvre it into difficult-to-get-to spots in your back garden.

How is it controlled right now?

At the moment, we're showing it off with a PlayStation controller, connected over Bluetooth. We have autonomous software so it will control itself, and we have additional boards that we can use to make it navigate a certain direction; for example, keep going south. We'll be adding cameras and sensors in the coming weeks and making this a fully autonomous robot.

How much torque does it generate?

They're 19-amp motors, 24 volts. So their maximum

"We have autonomous software so it will control itself, and we have additional boards that we can use to make it navigate a certain direction"

Below The all-terrain tyres allow the DoodleBorg to go everywhere. It seems that nowhere is safe!

theoretical varies at 2.1 KW – which is approximately three horsepower. That should be about enough to pull a caravan.

Will this just be a hobby project, or do you plan to sell it as a kit?

It would probably be a bit expensive as a kit, although we can build them if someone wants. This has cost us probably over £1,000. [Which is probably not all that much once you've taken into account] all the laser cut and the size of the bearings there. It is fairly bulletproof, though. While we won't be selling the robot itself as a kit, we are definitely selling the little PicoBorg reverse boards.

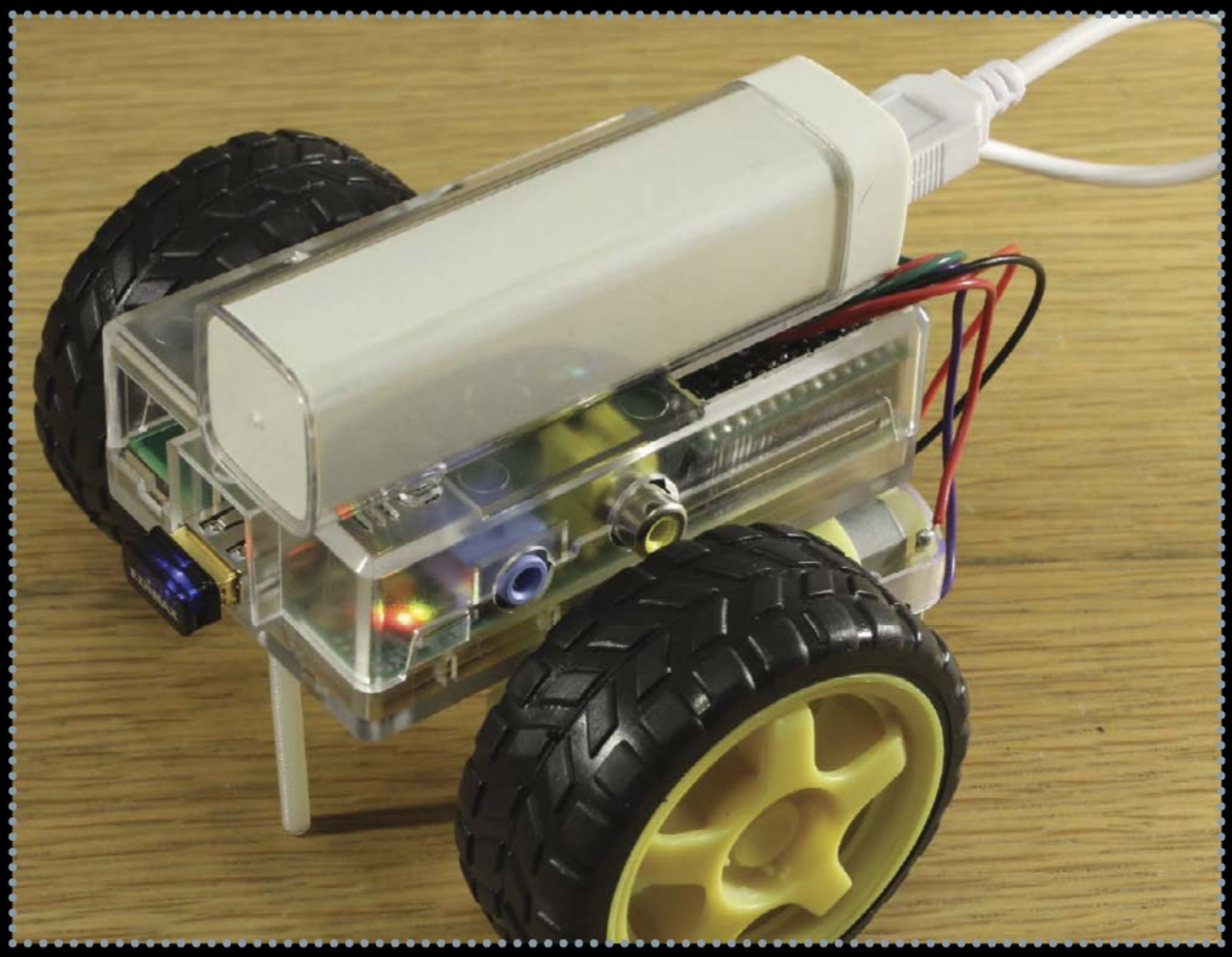
When will these PicoBorgs be available for sale?

They're available at the moment for pre-order on the website. We'll have a run of 250 of them and that should be available [by the time this magazine has been released]. This board will control small motors, it will control stepper motors, it will control big motors. They're all forwards and reverse with VWMs – speed control. Also, you can keep stacking them up, so you can add an extra axis, an extra motor, an extra anything. So, for example, CNC machines you might have two stepper motors to get your XY stage, and then have one for heat and that sort of thing. It's very scalable.

How much interest has the DoodleBorg got?

Lots of interest; [at the February CamJam] it appealed to both the kids and adults. I think DoodleBorg can be a little intimidating but once you get above a certain height, it's probably okay, and then there's more interest in this. But we've certainly had a lot of people playing with the little PiCy packs.

"You can keep stacking (PicoBorgs) up, so you can add an extra axis, an extra motor, an extra anything"



Left The PiCy is the DoodleBorg's tiny little brother, and can be controlled in much the same ways

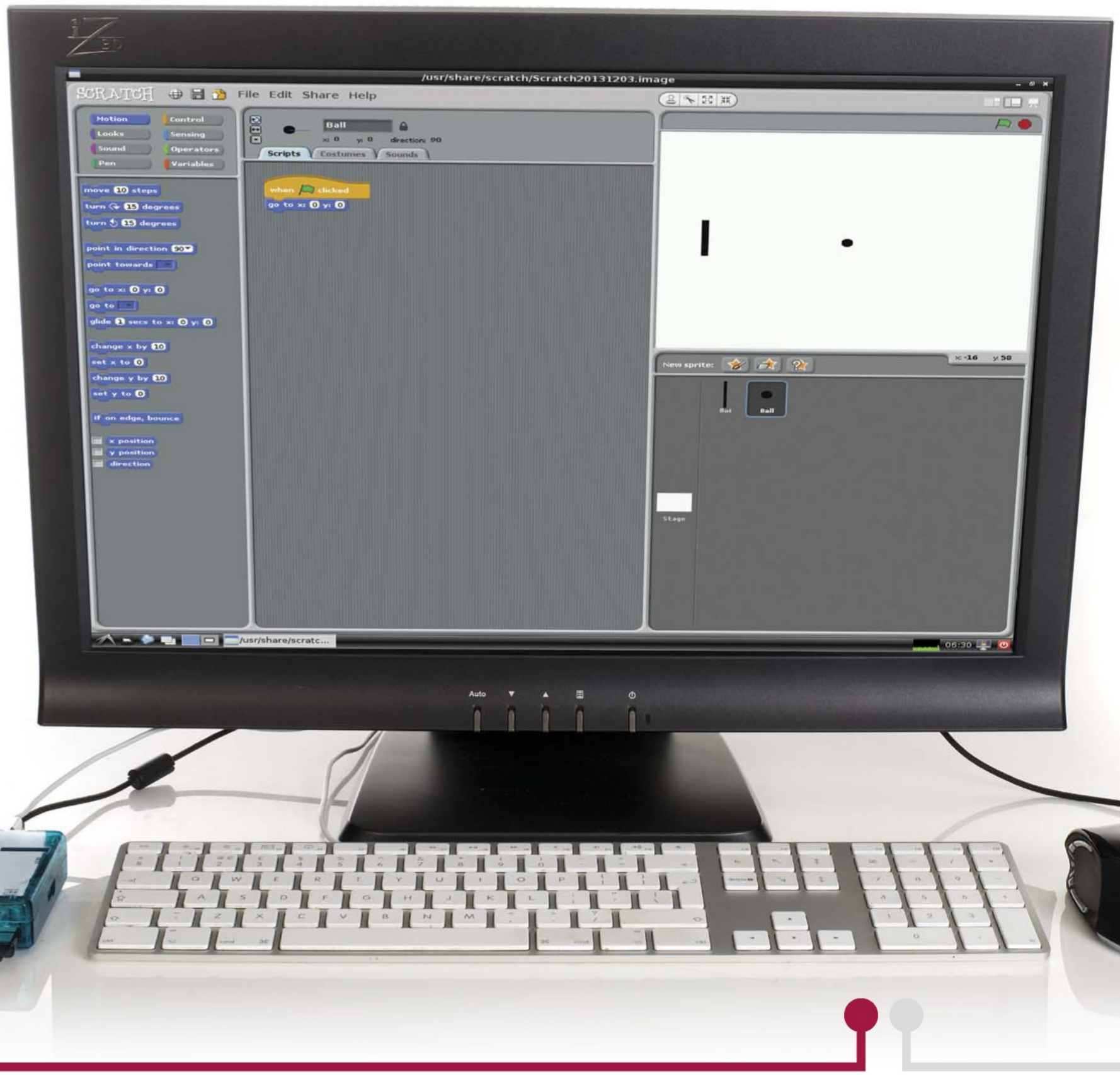
What's in the future for PiBorg?

In the future we do have some things coming up; at the moment we have a PicoBorg, which is an inexpensive on-off switch. The PicoBorg Reverse is the more expensive version. We have the XLoBorg, which is an accelerometer and magnetometer that you can use as a compass. We have LedBorg, an RGB LED, and that's a great way to learn how to do some further set up on the Raspberry Pi. And we also have our TriBorg which all of these mount on. We're basically moving at the movement to doing purely just Raspberry Pi products – and potentially some Arduino-based control things in the future.

"We have LedBorg, an RGB LED, and that's a great way to learn how to do some further set up on the Raspberry Pi"

Create a simple game with Scratch

Learn the basics of coding logic by creating a squash-like Pong clone in Scratch that you can play



While Scratch may seem like a very simplistic programming language that's just for kids, you'd be wrong to overlook it as an excellent first step into coding for all age levels. One aspect of learning to code is understanding the underlying logic that makes up all programs; comparing two systems, learning to work with loops and general decision-making within the code.

Scratch strips away the actual code bit and leaves you with just the logic to deal with. This makes it a great starting point for beginners, separating the terminology so you can learn that later on when you choose to make a proper program. Scracth is also included within every copy of Raspbian.

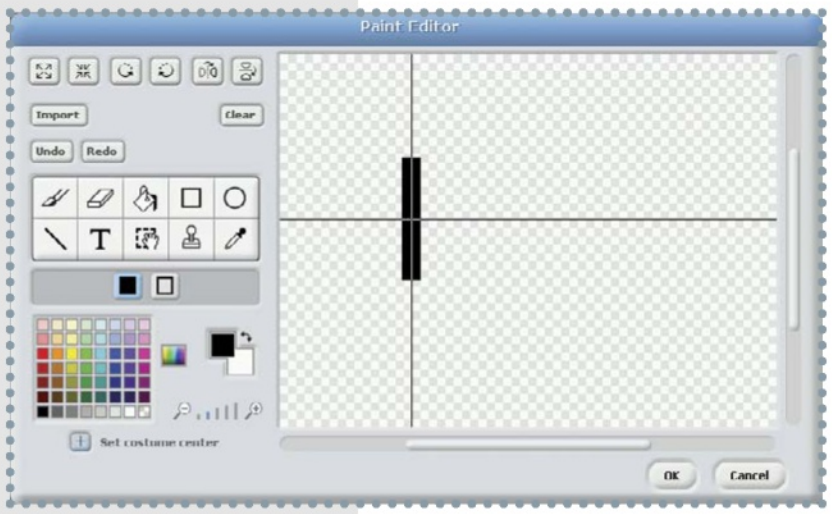
> "Scratch strips away the actual code and leaves you with just the logic – this makes it a great starting point for beginners"

01 The first sprite

Opening up Scratch will display a blank game with the Scratch cat; right-click on it and delete to remove it. Click the Paint New Sprite button below the game window and draw a slim rectangle for the bat using the square drawing tool. Click Set Costume Center so that Scratch knows the basic dimensions of your bat; drag it to the centre of the sprite if needs be.

02 Move the bat

Click OK, and name the new sprite 'Bat'. Select the Control block from the top-left menu, and drag the When Space Key Pressed block into the script area. Change Space to Up Arrow from the drop-down menu, then select the Motion options and drag the Change Y By 10 block to connect to the key pressed block.



Below We'll use this bat sprite to interact with the ball sprite

03 Reverse the direction

Whenever we press up, the bat will move up by ten pixels, as set. By repeating the process using the down arrow and setting Change Y To -10, we can also make it move down as well. Move the rectangle over to the left-hand side of the screen. This will be our starting position.

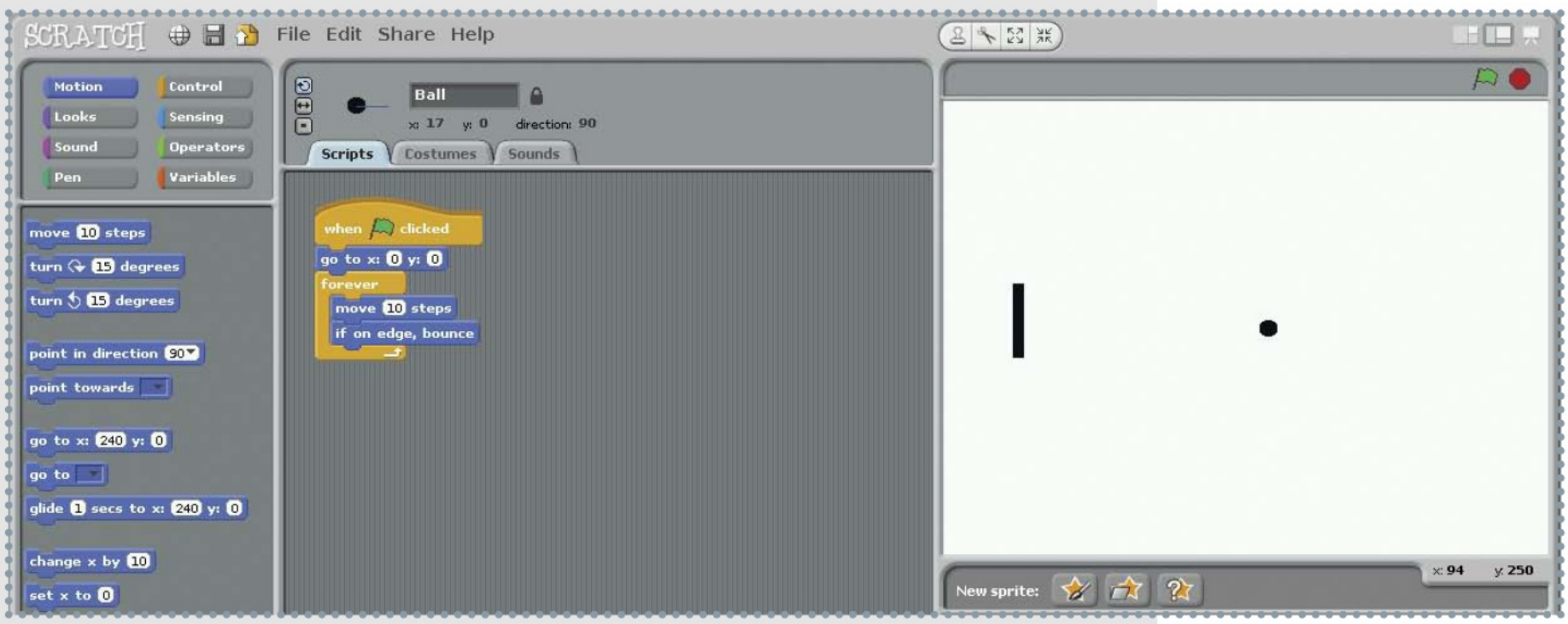
04 Create the ball

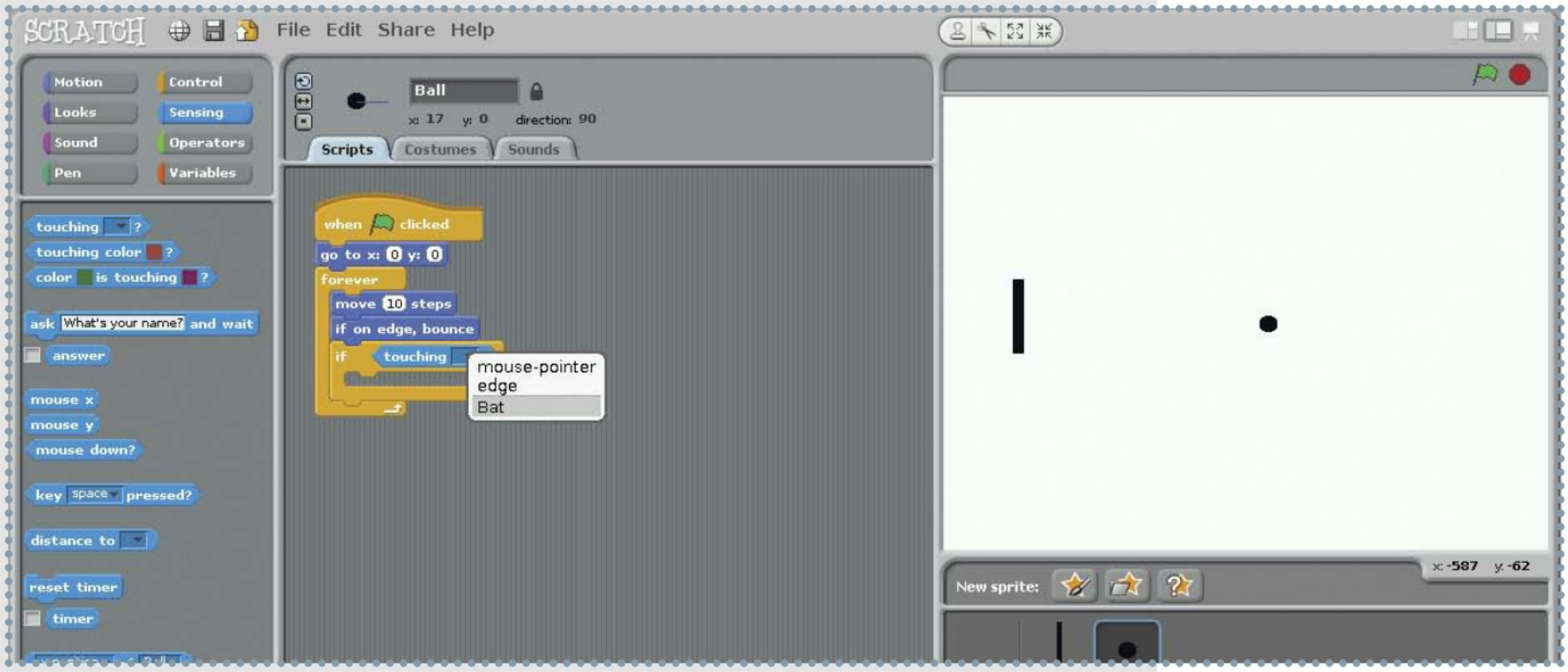
Create a sphere from the new sprite menu like we did with the bat, including setting the Costume Center. Name it 'Ball', and bring the When Green Flag Clicked block from the Control menu into the scripts pane. Add the go to x:0 y:0 block underneath it so that whenever you click the green flag it resets to the ball.

05 Move the ball

Find the Forever block in the control menu and attach it under our existing block on Ball. Then, go back to the Motion menu and add Move 10 Steps so that the ball moves around the screen. Add the If On Edge, Bounce block from Motion below that so that it will stay within the playing field.

Below Edge detection will ensure our ball doesn't just disappear when it hits a border





Above The block library in the left pane is broken down into simple categories

06 Hit the ball

Drag the If block from Control to below If On Edge, and then add Touching from Sensing to the empty space on the If block. From the drop-down menu, select Bat so that it will interact with our bat sprite. Add one of the Turn 15 degrees blocks from Motion to the If block, and change it to 180.

07 Simple bouncing

Pressing the green flag now, the ball will bounce between the left and right edges of the screen, or off the bat if it comes into contact with it. We can make it slightly more interactive to make use of the moving bat, and more like Pong.

08 Random bouncing

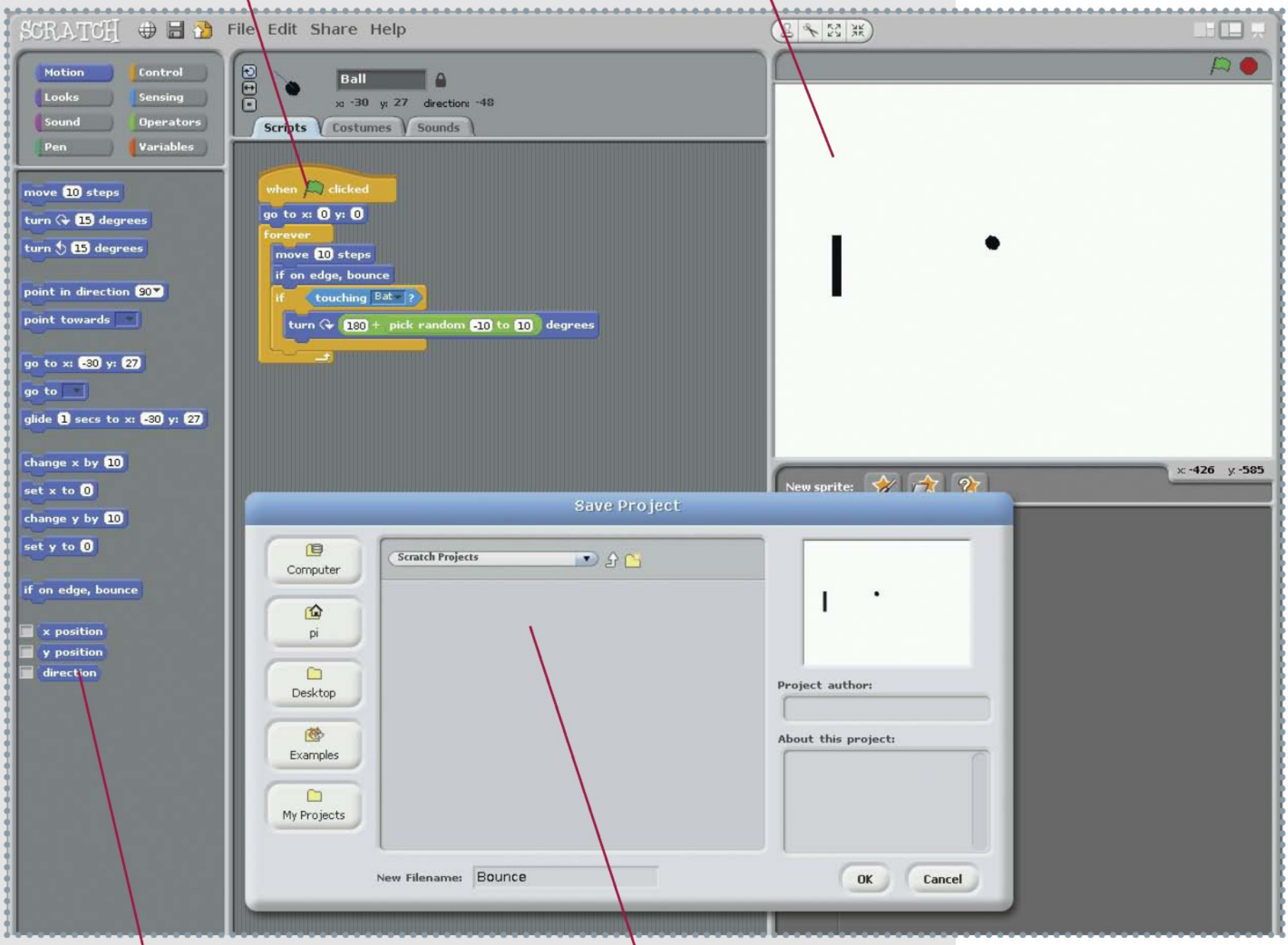
Go the Operators menu and select the first value, blank + blank. Drag it to where we have 180 degrees in the turn block, and add 180 to the first blank space. Place the Pick Random 1 to 10 block in the second space and change the values to -10 and 10 so that whenever the ball hits the bat, it bounces off at a random angle between 170 and 190 degrees.

Building blocks

Use blocks that represent coding to build your game or animation

Preview

Test your code straight away to make sure it does what you expect it to do



Code logic

Slot the blocks together nice and easily to create loops and comparisons

Build project

Export to the Scratch website to show off your work to the world

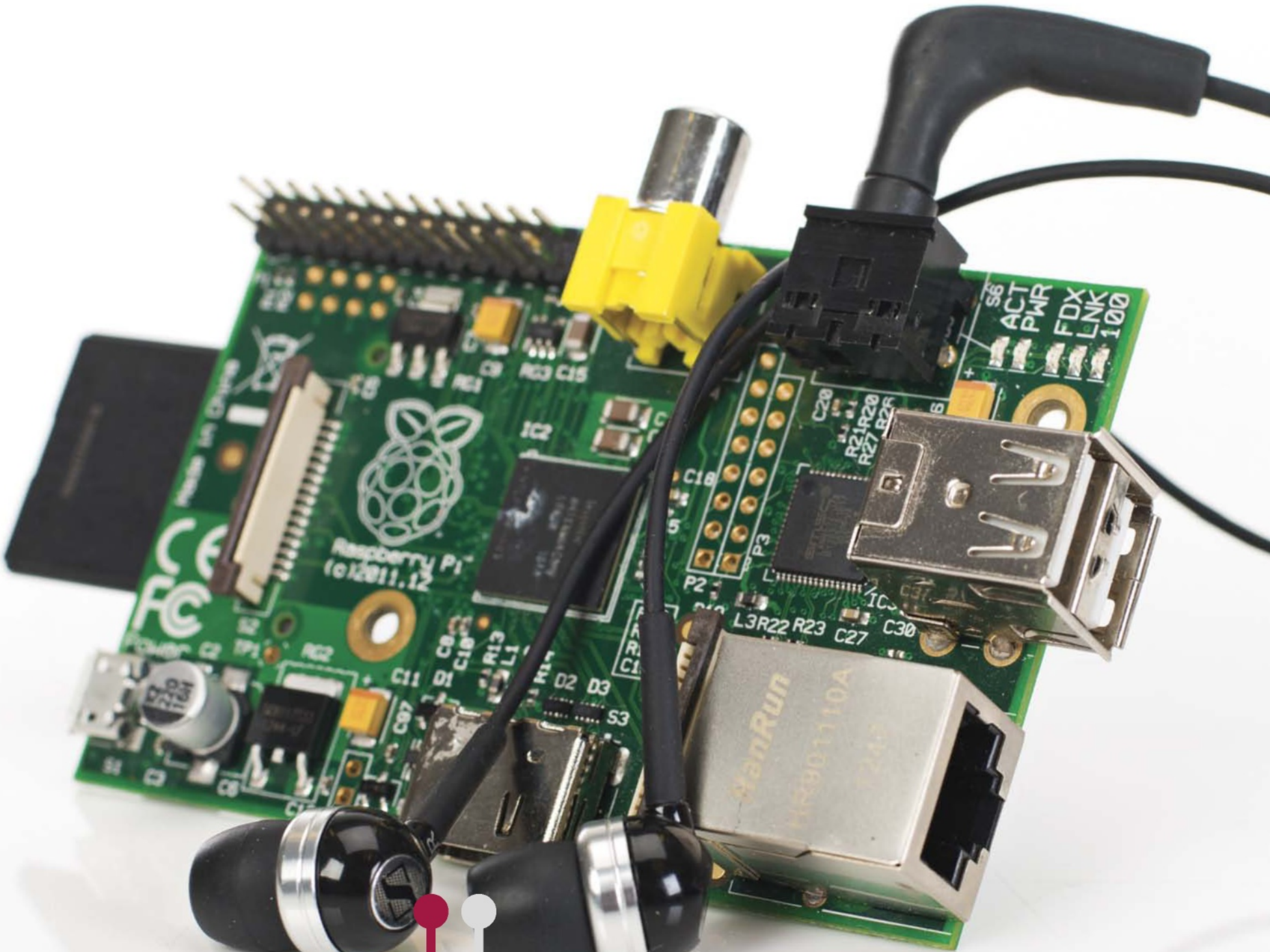
09 Further developments

While you now have a functional game of sorts, you can also add in a second player to make it truly Pong-like, and add a scoring system by having a number increase when the ball hits one of the sides.

"Test your code straight away to make sure it does what you expect it to do"

Learn to code with Sonic Pi

Take the next step in programming and create your own melodies with Sonic Pi, the musical programming language



With Scratch we've learned how to operate under the logic of programming. The next step is to then use that within a programming language – the problem is that many of the available languages can look a little intimidating. This is where Sonic Pi comes in, offering a very simple language style that can ease you in to the basics of working with code. It's quite straightforward to use as well – Sonic Pi allows you to choose from a small selection of instruments and select a tone to play with it. These can be turned into complex melodies using loops and threads and even some form of user input.

THE PROJECT ESSENTIALS

Speakers or headphones

Sonic Pi

http://www.cl.cam.ac.uk/projects/raspberrypi/sonicpi/

01 Install Raspbian

If you've installed the latest version of Raspbian, Sonic Pi will be included by default. If you're still using a slightly older version, then you'll need to install it via the repos. Do this with:

```
$ sudo apt-get install sonic-pi
```

02 Get started with Sonic Pi

Sonic Pi is located in the Education category in the menus. Open it up and you'll be presented with something that looks like an IDE. The pane on the left allows you to enter code, and then you can save and preview it as well. Any errors are displayed separately from the output.

> "Sonic Pi allows you to choose from a small selection of instruments and select a tone to play with it. These can be turned into complex melodies"

03 Your first note

The first thing to try out with Sonic Pi is simply being able to play a note. Sonic Pi has a few defaults already pre-set, so we can get started with:

```
play 50
```

Press run and the output window should show you exactly what is happening.

04 Set the beat

For any piece of music, you'll probably want to set the beat. We can start by putting:

```
with_tempo 200
```

At the start of our code. We can then test this out by creating a string of midi notes using play_pattern.

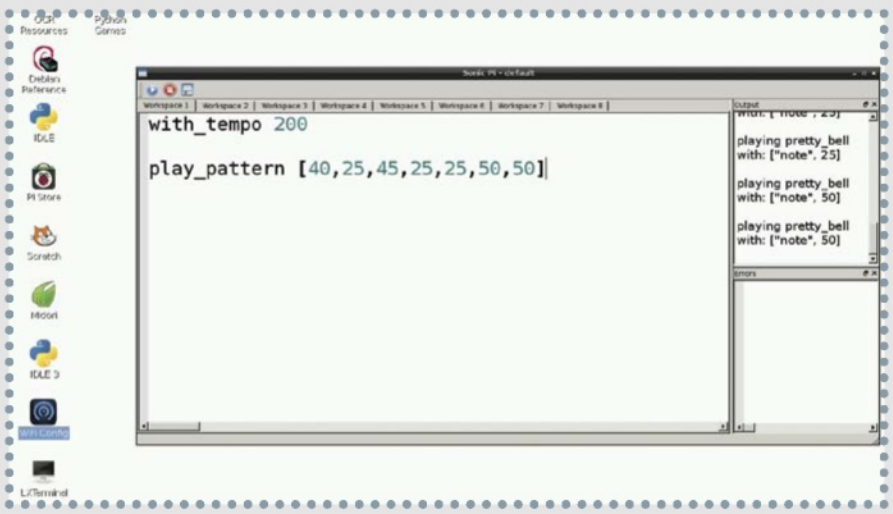
05 Advance your melody

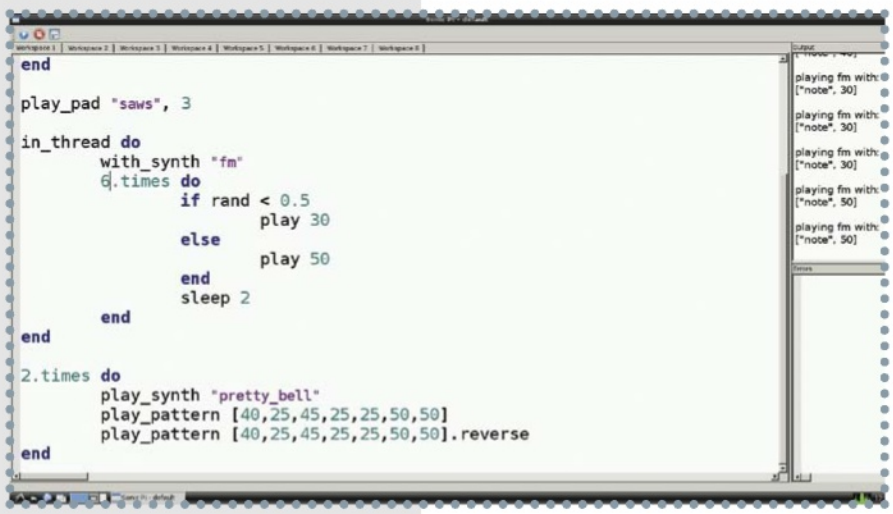
We can start making more complex melodies by using more of Sonic Pi's functions. You can change the note type by using with_synth, reverse a pattern, and even create a finite loop with the x.times function. 'Do' and 'end' signify the start and end of the loop.

06 Play a concert

Using the in_thread function, we can create another thread for the Sonic Pi instance and have several lines of musical code play at once instead of in sequence – check out the lower-right screenshot, just here:

"Change the note type by using with_ synth, reverse a pattern, and even create a finite loop with the x.times function"





Below Left You can test the with_tempo command by playing a string of notes with play_pattern

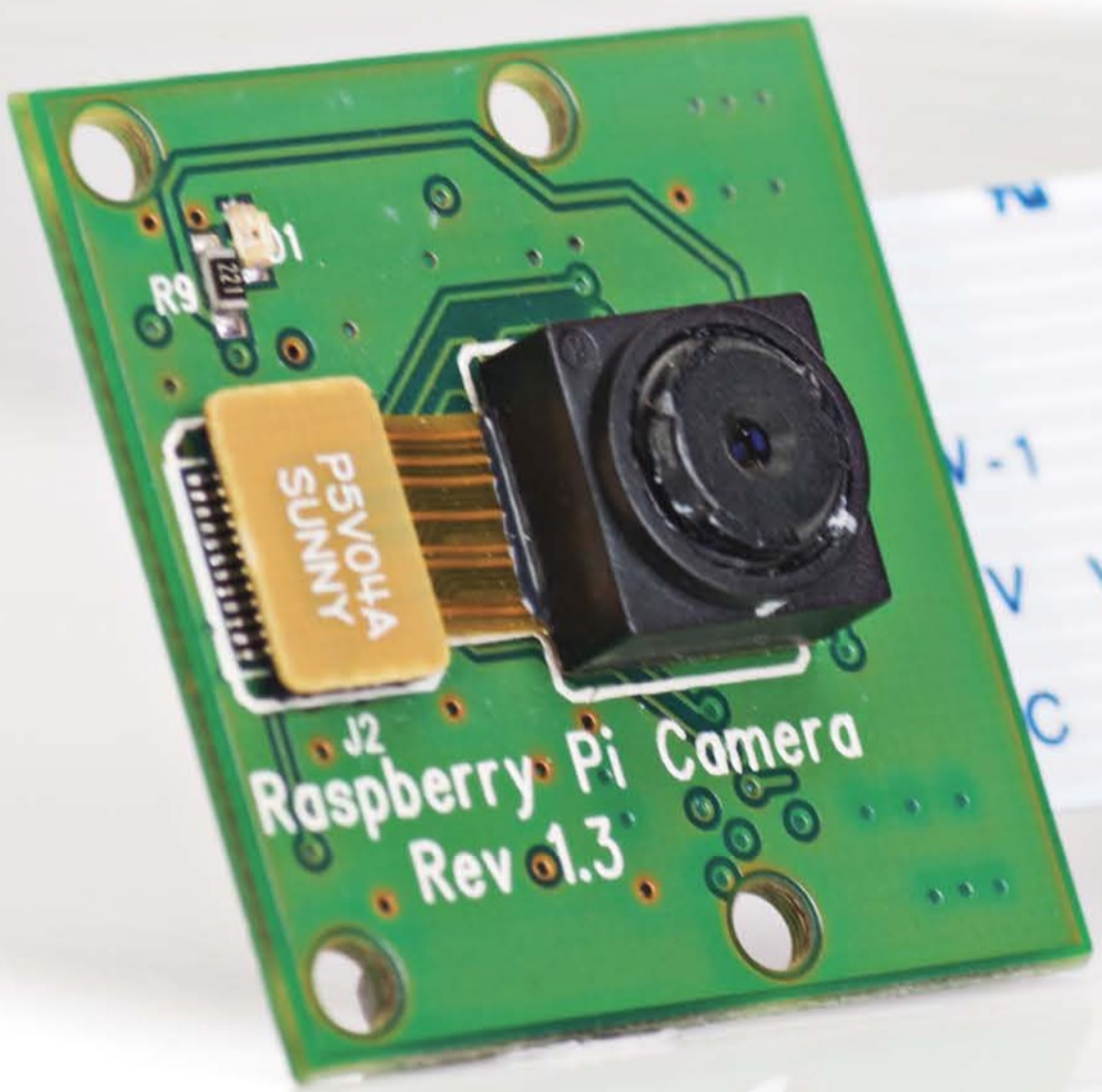
Below Here we've made Sonic Pi create a series of notes in a random sequence

The Code COMPOSE YOUR FIRST SONIC PI MELODY

```
with_tempo 200

play_pattern [40,25,45,25,25,50,50]

2.times do
    with_synth “beep”
    play_pattern [40,25,45,25,25,50,50]
    play_pattern [40,25,45,25,25,50,50].reverse
end

play_pad “saws”, 3

in_thread do
    with_synth “fm”
    6.times do
        if rand < 0.5
            play 30
        else
            play 50
        end
        sleep 2
    end
end

2.times do
    play_synth “pretty_bell”
    play_pattern [40,25,45,25,25,50,50]
    play_pattern [40,25,45,25,25,50,50].reverse
end
```

Take photos with Raspberry Pi

Use the Raspberry Pi camera module to capture photos and video, so you have a portable camera for any situation



The Raspberry Pi camera module is an excellent addition to the Raspberry Pi. Not only does it slot into one of the non-traditional ports on the board itself, but it's also easily programmable within Raspbian. This gives it a few benefits over a USB webcam by not taking up any USB slots and being easier to control with code. It's also tiny, making it as portable as the Raspberry Pi itself.

By the time you reach the end of this tutorial, you'll be able to use the camera like a pro to create time delays and specific image formatting.

"Not only does it slot into one of the non-traditional ports on the board itself, it's also easily programmable in Raspbian"

THE PROJECT ESSENTIALS

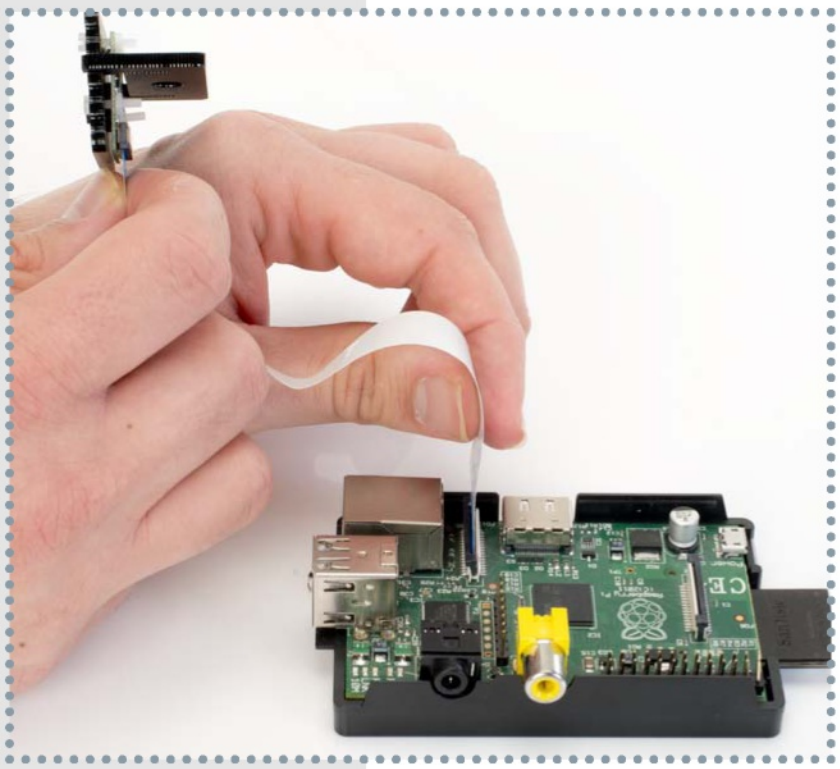
Raspberry Pi camera board picamera

github.com/waveform80/picamera

01 Plug in your camera

To attach the camera to the Raspberry Pi, locate the connectors between the ethernet and HDMI port and gently lift up the fastener. Insert the camera board ribbon, with the metal connectors facing away from the ethernet port.

Below Make sure you keep the metal connectors facing away from the ethernet port



02 Enable the camera

First you'll need to make sure the camera modules are enabled. To start the standard configuration screen, open a terminal and type:

```
$ sudo raspi-config
```

Navigate down to Enable Camera, press Enter, and then simply key over to enable and confirm with another press of Enter. Select Finish and then reboot.

03 Take your first picture

Capturing pictures with the Raspberry Pi Camera is straightforward, all you have to do is enter:

```
$ raspistill -o image.png
```

This command will show you a five-second preview of the input of the camera and then capture the last frame of the video.

04 Record some video

To record a video, we use a similar command, raspivid, like so:

```
$ raspivid -o video.h264
```

Just like with the image-capturing that we did in the previous step, this will display a preview of what the camera is seeing. However, the video actually records the five seconds that make up the preview as well.

05 Advanced Pi camera uses

If you want to be able to do a little more with the camera, there's a simple Python wrapper currently available called picamera. You'll need to install it first though, and you can do so from the terminal, using the following command:

```
$ sudo apt-get install python-picamera
```

06 Python test

Let's make sure that everything we've done still works. Enter a Python shell by typing 'python' into the terminal, and then type the following three lines:

```
import picamera
camera = picamera.PiCamera()
camera.capture('pythontest.png')
camera.close()
```

> "If you want to do a little more with the camera, there's a simple Python wrapper called picamera"

Below If your terminal looks like this then you're doing everything right

```
pi@raspberrypi ~ $ sudo apt-get install python-picamera
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  python-picamera-docs
The following NEW packages will be installed:
  python-picamera
0 upgraded, 1 newly installed, 0 to remove and 1 not upgraded.
Need to get 53.2 kB of archives.
After this operation, 302 kB of additional disk space will be used.
Get:1 http://archive.raspberrypi.org/debian/ wheezy/main python-picamera arm
Fetched 53.2 kB in 0s (345 kB/s)
Selecting previously unselected package python-picamera.
(Reading database ... 68995 files and directories currently installed.)
Unpacking python-picamera (from .../python-picamera_1.2-1_armhf.deb) ...
Setting up python-picamera (1.2-1) ...
pi@raspberrypi ~ $ python
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import picamera
>>> camera = picamera.PiCamera()
>>> camera.capture('pythontest.png')
>>> camera.close()
>>>
```

07 Test explained

Press Ctrl+D to exit the Python shell. We just used code similar to the command line tools to take a simple image called 'pythontest.png'. The most important thing we did after that was 'camera.close', to make sure that the camera was turned off after use.

08 Python video

To record video with picamera, you need to first set the resolution and then set a recording time.

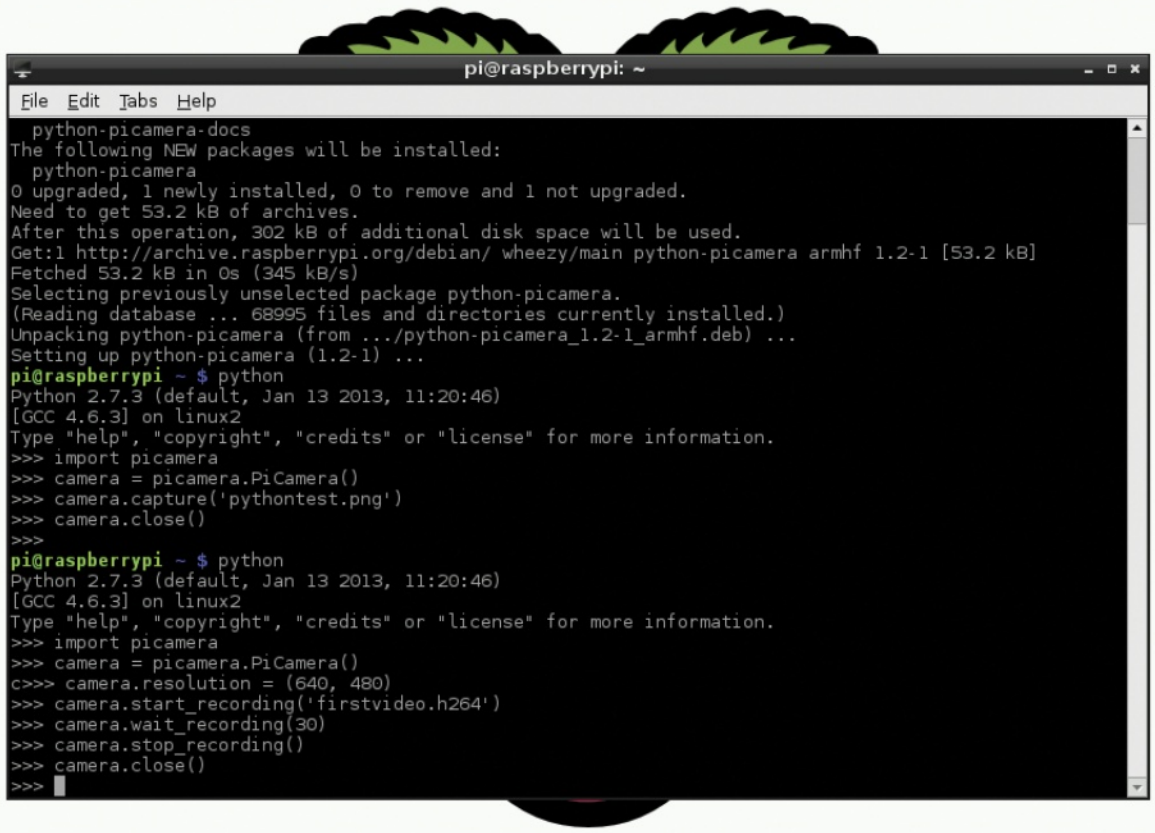
```
import picamera
camera = picamera.PiCamera()
camera.resolution = (640, 480)
camera.start_recording('firstvideo.h264')
camera.wait_recording(60)
camera.stop_recording()
camera.close()
```

09 More code

The above works a lot better in a Python script – the wait is only really required if you can't manually stop the recording.

The picamera module allows you to create time lapses, modify the frame rate of recordings and much more. For more ways to use it, check out the official documentation on the picamera GitHub page.

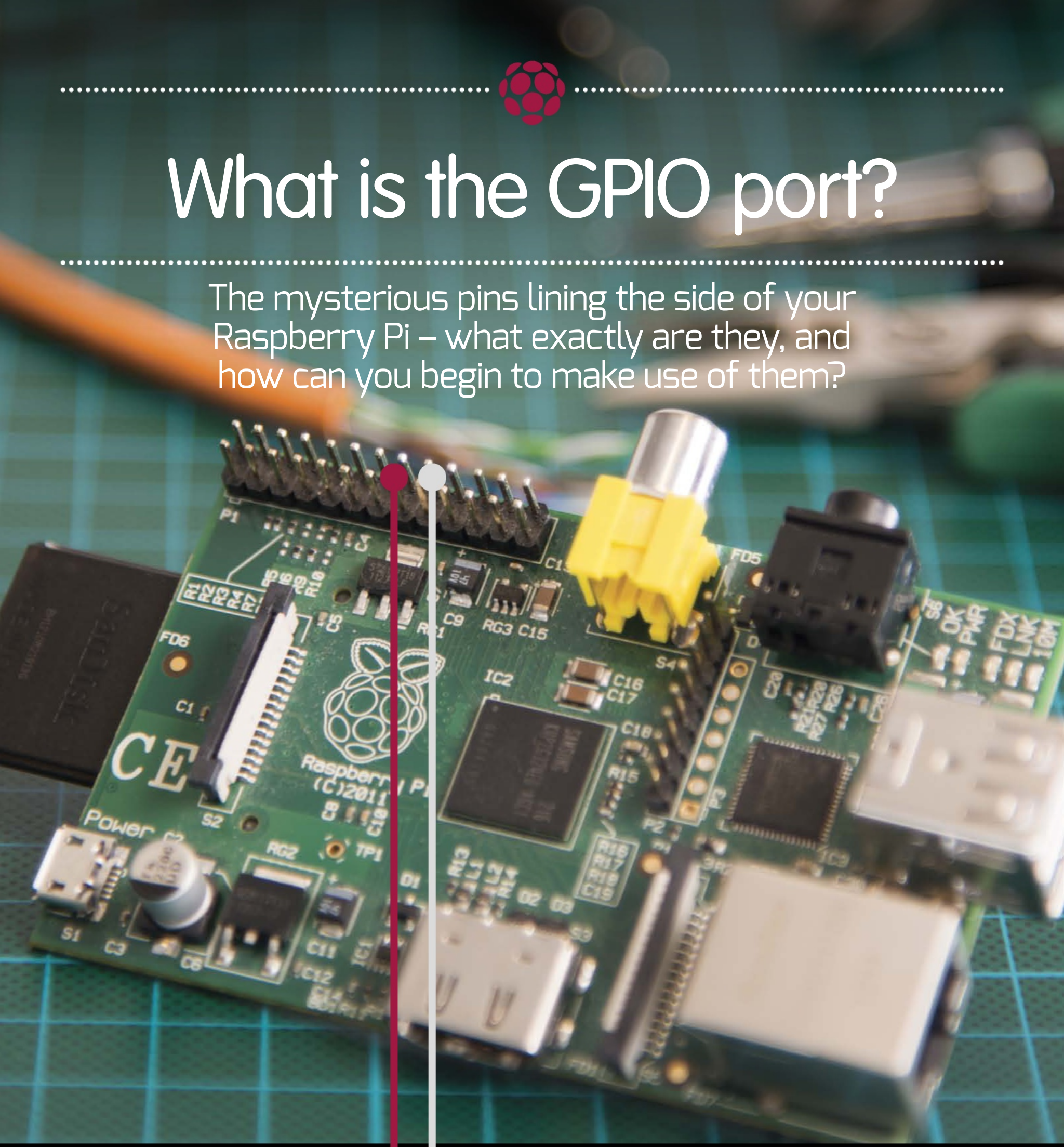
"The picamera module allows you to create time lapses, modify the frame rate of recordings and much more"



Below Remember to turn the camera off with camera.close() once you're finished

What is the GPIO port?

The mysterious pins lining the side of your Raspberry Pi – what exactly are they, and how can you begin to make use of them?



"The GPIO port exposes the CPU inputs and outputs to anything properly connected to it"

Q A lot of the ports and outputs and such are quite obvious on the Raspberry Pi, but there are also a few extra pins and connectors on the board itself. Are these actually usable?

A Very much so; there are three individual low-level peripherals on the Raspberry Pi that can be used by the Raspberry Pi itself. The two white ports and the large strip of pins on the edge next to the yellow Video-out port are all usable on all versions of the Raspberry Pi.

Q Okay, good to know. What about the other set of pins between the video and audio port?

A These are the testing connectors used during development of the board. These do not work on newer versions of the Raspberry Pi, but they originally connected to the VideoCore chip on the board. You can pretty much ignore them if you like.

Q Fair enough. So what do these functioning ports do?

A The white connector next to the ethernet port is a CSI-2 interface; this takes video input and is used by the camera module. The other white connector is the DSI interface, which can output display data if it's hooked up correctly. The final set of pins are what's known as the GPIO ports.

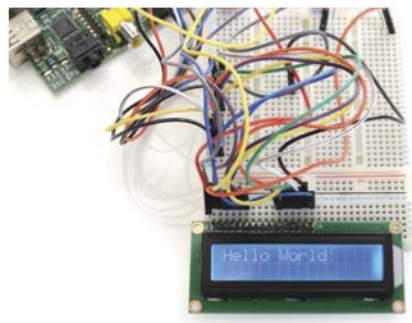
Q I've heard the term GPIO come up a lot with Raspberry Pi stuff. What does GPIO stand for?

A GPIO is a generic term that stands for General-Purpose Input/Output; it's not just the name for what's on the Raspberry Pi. Many systems will have GPIO ports you can use.

Does that mean that it uses a standard set of pins then?

A No, there aren't any standards for what makes up a

What you can do...



LCD display
Get started with a simple project that lets you display text on an LCD screen

Time-lapse photography
Go a bit further and have it interact with a consumer product – connect it to a camera and create stop-motion videos

"GPIO is a generic term that stands for General-Purpose Input/Output"

series of GPIO pins or port. This layout is specific for the Raspberry Pi in particular.

Q If it's not a standard, then how do you know which pins do what?

A There are handy diagrams explaining what each pin does, and a lot of projects will tell you exactly how to wire it up if needs be.

Q So some projects are using these GPIO pins then? How exactly can you make use of them?

A The GPIO port exposes the CPU inputs and outputs to anything properly connected to it. You can do things as simple as sending an on/off signal for a light, or receive and send data through the pins. Most projects that use the Raspberry Pi do more than connect to the internet and display data; they could use it to control an automated house, a robot, an Arduino circuit, a weather station or anything in between. We've used it to create traffic lights, connect an accelerometer and even in RC cars and quadcopters.

Q So it's very easy to use then?

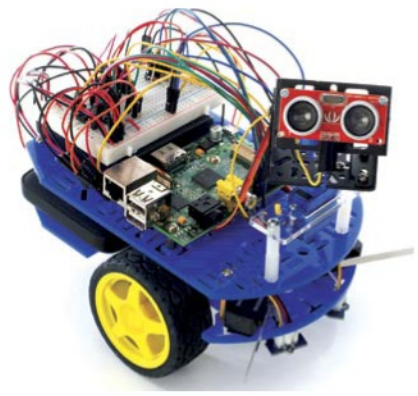
A Well, not exactly. To some people, yes, but it does involve using proper code to connect and talk to it. You can also use a Scratch-like graphical interface to control it if you wish. Once you know what it is you're doing, it can be as easy as turning on a light.

Q You mentioned Arduino circuits – is this the same thing the Gertduino and Arduberry board use?

A Yes, the Gertduino and Arduberry are two boards that expand upon the GPIO port in order to allow you to get more from your Raspberry Pi and also connect it to an Arduino single-board microcontroller. The Gertduino and

What you can do...

Add extras
Extra boards like the Gertduino use the GPIO ports to expand the Raspberry Pi's functionality



Robotics
The ultimate pay-off: build your own Raspberry Pi robot, controlling it via the GPIO inputs

"We've used it to create traffic lights, connect an accelerometer and even in RC cars and quadcopters"

Gertboard need to be wired up to the the GPIO port, while the Arduberry will sit on top of the Raspberry Pi itself.

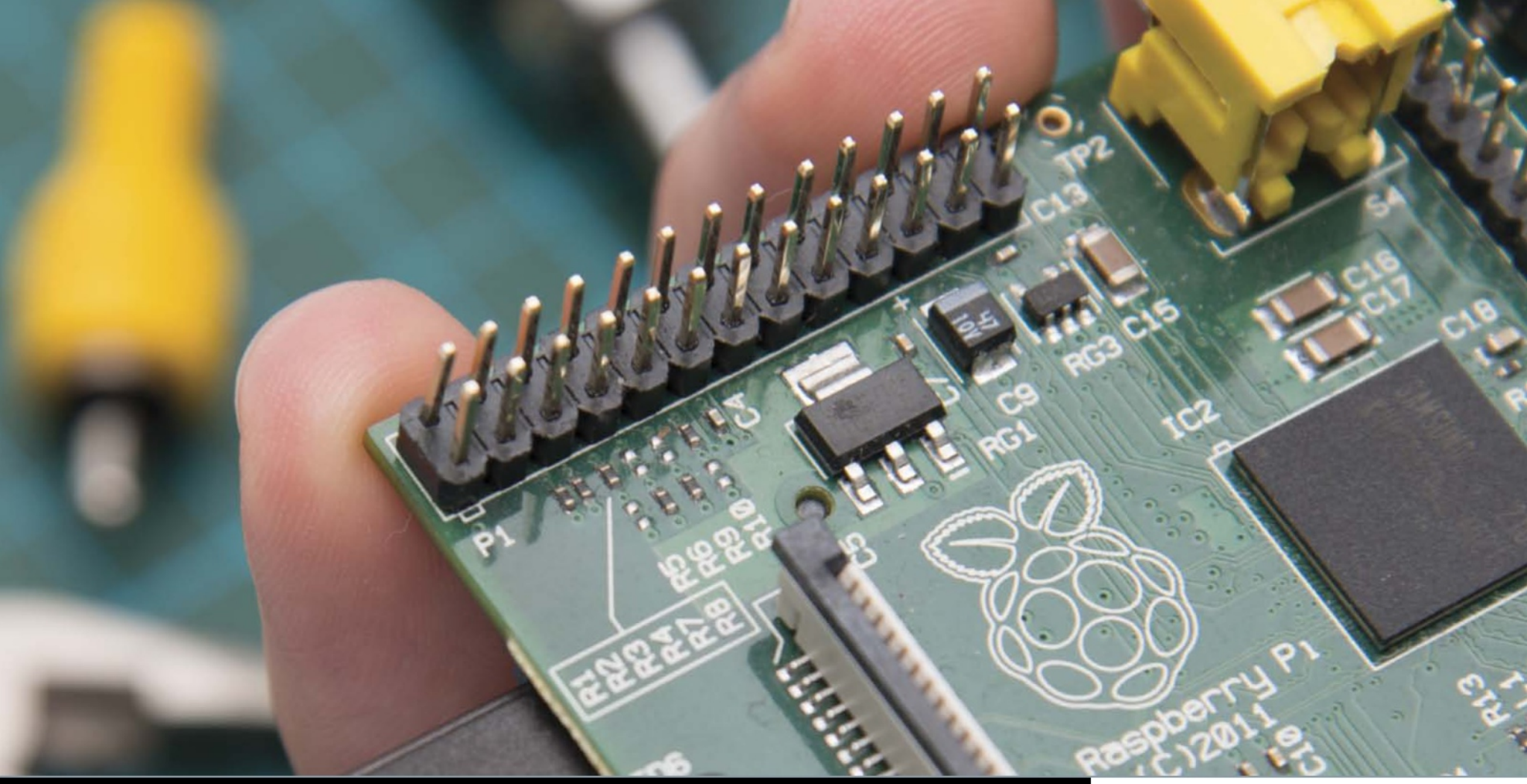
Q Why would you choose to have to wire-up a board rather than simply sit the other boards on top?
A Having the extra board on top means you won't have ready access to the other connectors we just mentioned, and it also wouldn't easily fit inside any standard Raspberry Pi case.

Q But doesn't that mean that you wouldn't actually be able to connect it to a Raspberry Pi in the first place, if it was in a case?
A Not necessarily; there are a number of cases that actually take the GPIO ports into account and allow the cables access to the pins. There are some that also allow access to the DSI and CSI interfaces, but they are a little rarer.

Q That's just reminded me of something else I've always wondered – isn't it quite dangerous for the Raspberry Pi to be a bare board like it is? Could these ports get damaged and cause all kinds of problems?
A Theoretically yes, the Raspberry Pi is a bit more susceptible to damage by not being supplied with any kind of case. However the board is pretty robust even without a case; as long as you don't let dust properly build up on it and take care of it, you should be fine.

Q Dust my Pi or put it in a case, gotcha. Well, where can I go to learn a bit more about the GPIO port? Are there any cool projects that will help me learn a bit more about them?
A Well first of all, we thought you might ask that and so we've got a GPIO tutorial coming right up that you can

"The (Raspberry Pi) is pretty robust even without a case, as long as you don't let dust properly build up on it and take care of it"



have a go at – just swipe over to the next page. Another great resource for Raspberry Pi tutorials is our sister magazine's website – just use the search field at the top-right of www.linuxuser.co.uk.

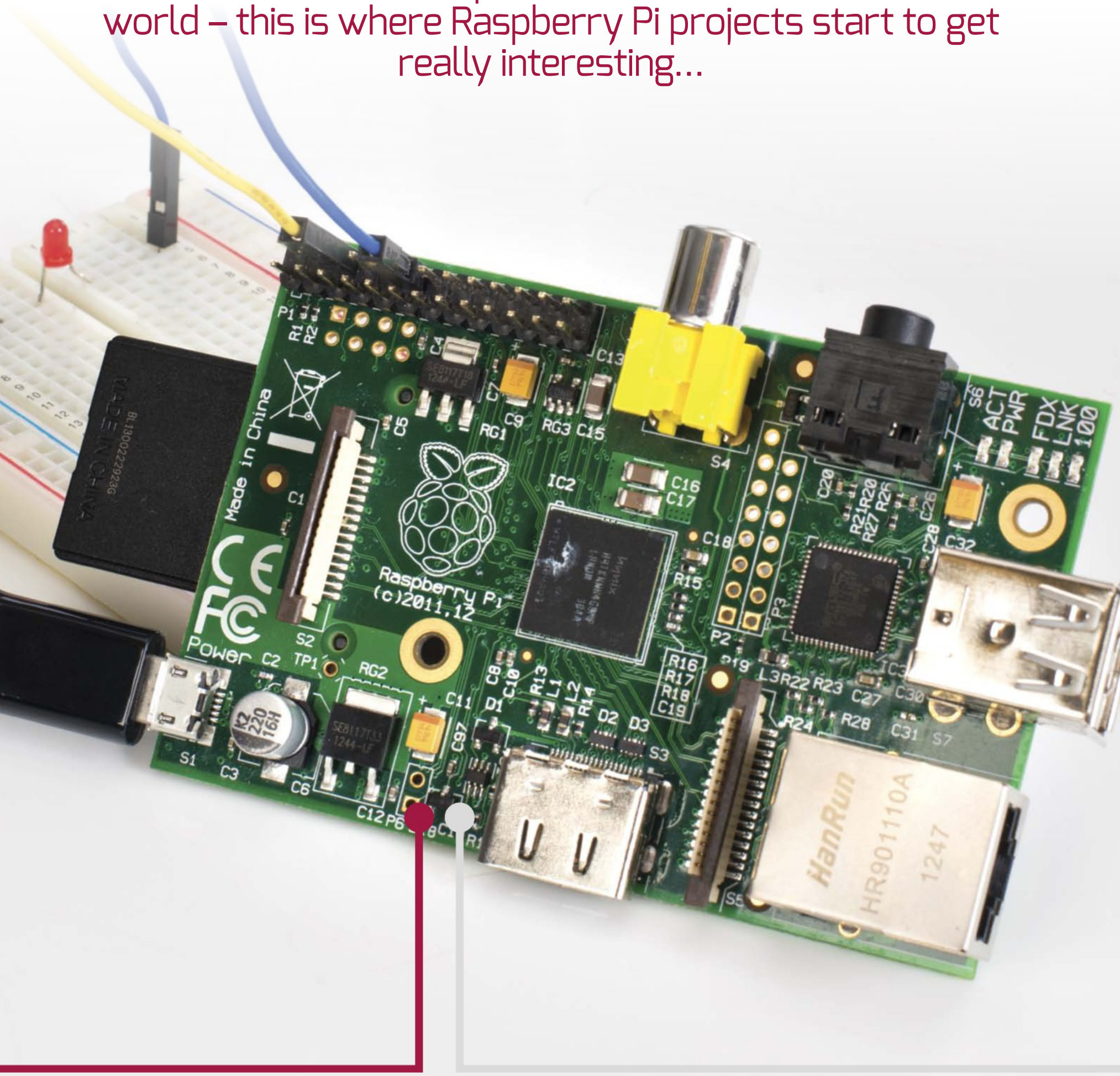
As for online resources, you should be able to find some OCR courses on creating simple projects with the GPIO ports. These are available for free on Raspbian from the desktop, although you can also find them on the OCR website available from: www.ocr.org.uk/raspberrypi. These include some simple but really fun projects such as traffic lights, singing jelly babies and many more. Enjoy!

Below Use the GPIO ports to connect your Pi to all kinds of devices and interact with them

"You should be able to find some OCR courses on creating simple projects with the GPIO ports"

Use the GPIO pins

Learn to use the GPIO pins to interact with the outside world – this is where Raspberry Pi projects start to get really interesting...



The GPIO pins give you power to interact with the real world using your Raspberry Pi. This project will get you comfortable with using the GPIO pins, which form the backbone of many projects.

Traditionally the 'Hello World' program for electronics prototyping is simply turning a light on and off. We're going one better than that by simulating candlelight. To do this we'll use the Random and Time modules from Python's standard library to continually change the brightness of the light while using the RPi-GPIO library to control the LED with Pulse Width Modulation (PWM).

THE PROJECT ESSENTIALS

Small breadboard

Small LED (any colour)

330 Ohm resistor

2x male-to-female prototyping cables

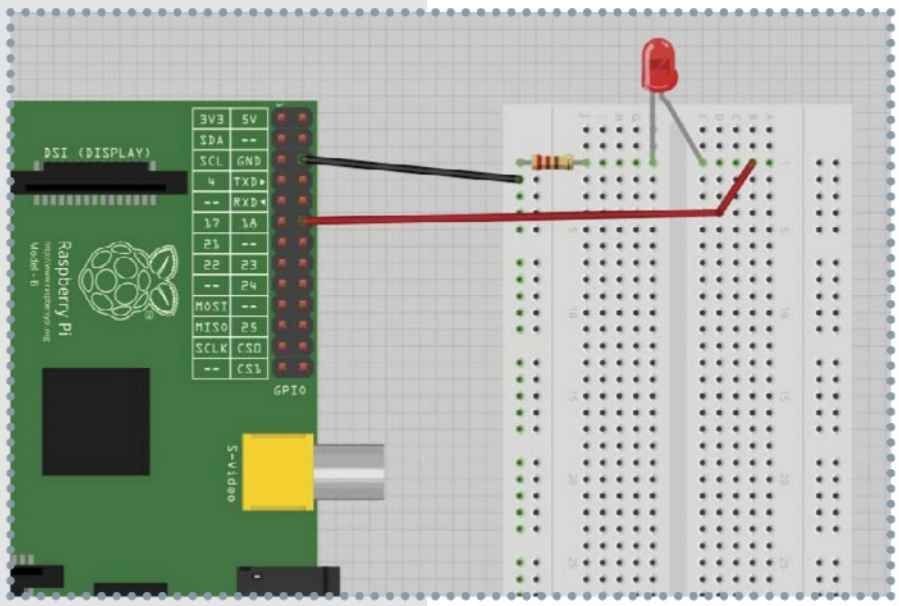
01 Prepare your circuit

Before we turn on our Raspberry Pi we'll make all the physical cable connections to it. This requires us to place a male-to-female cable on the first ground pin on the GPIO port and another male-to-female cable on the PWM pin (pin 18). Connect it to the breadboard along with the LED and resistor as shown, ensuring the short leg of the LED goes to ground.

02 Power up the Pi

With the circuit complete we can power up the Pi. Open Leafpad and we'll start creating the script (listed in a few pages) we need to control the LED light. The first thing we do is import the modules we need. The RPi.GPIO library is key to our project – we use it to read and write to pins and control their functionality. We're also using Random and Time modules in order to help simulate the effect of candlelight.

Below Hook up your LED and resistor to pin 18 (the one on the outside of the board)



03 Configure the GPIO module

Next we assign a name to the LED pin and set up the GPIO module for our project. Notice we're using 'setmode' and calling it BCM. This means we're using the Broadcom naming scheme. We then assign the LED pin to OUTPUT, which means we'll be outputting to that pin (as opposed to reading from it). If we simply wanted to turn the light on and off, at this point we could use GPIO.output(led,GPIO.HIGH) and GPIO.output(led,GPIO.LOW). Instead we're using PWM, so we assign a variable PWM to control it.

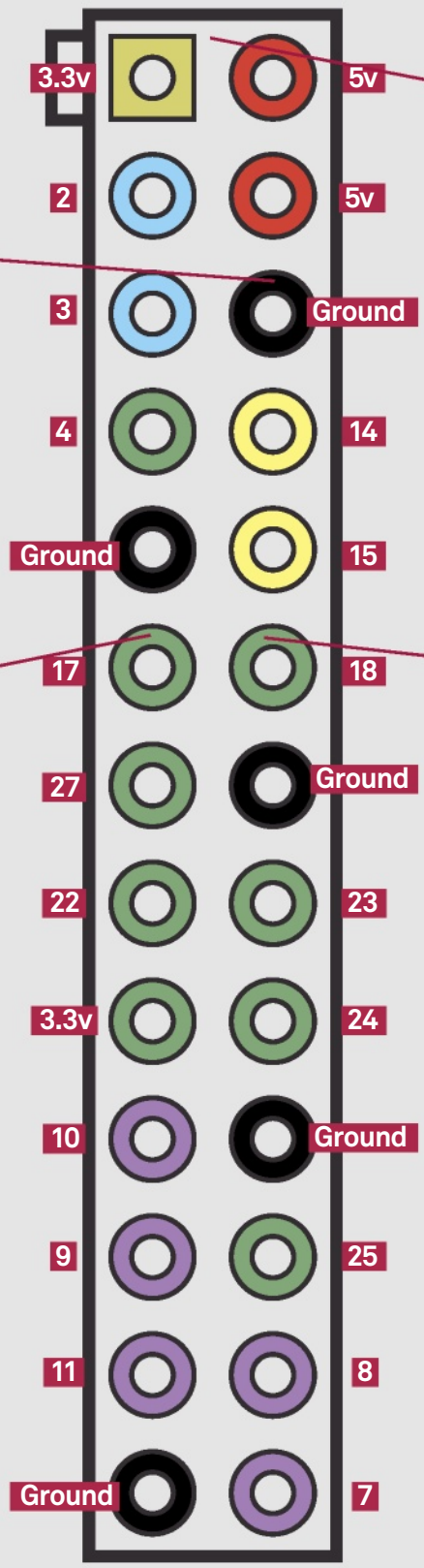
"The top of the image represents the end of the GPIO pins nearest to your SD card"

Grounded

The pins are counted from left to right and top to bottom. This is one of the few Ground pins, which we'll be using

BCM for Broadcom

We're using the BCM pin naming scheme for our projects. It doesn't use the physical pin locations, so BCM pin 17 is actually pin 11 when counted from left to right



This is the top!

The top of the image represents the end of the GPIO pins nearest to your SD card

Special pins

While most pins can be manually set to be outputs or inputs for any use, some pins are pre-assigned other roles too. This is the PWM pin, capable of controlling motors and LEDs with amazing precision

04 Basic functions

Next we add two functions to call in our program loop to randomly control the physical brightness of the LED and the amount of time that the light pauses on a set brightness. To do this we first use the random.randint method. The numbers 5 and 100 represent the lowest and highest brightness – the function will then pick a number between these percentages during each loop through the program. random.random picks a floating point number between 0 and 1 – we divide this by WIND to help achieve our flicker effect.

05 The main loop

We use a while loop to activate the light with PWM and change the brightness with the ChangeDutyCycle method, which calls our brightness function. Similarly, with time.sleep we get the brightness to maintain for a short, random amount of time by calling the flicker function. When we want to quit this otherwise infinite loop, we can raise a KeyboardInterrupt by pressing Ctrl+C. When we do, we need to free up the pins so they can be used again, by calling pwm.stop and GPIO.cleanup respectively.

06 Test your project

Once your script has been written, save it as candlelight.py. To run it, open the Terminal and type:

```
sudo python candlelight.py
```

You need to call sudo here since you need admin privileges to access the GPIO port on your Pi. Assuming you've set up the breadboard and copied the script correctly, your LED light should start flickering as if it's a candle in a light breeze. Take some time to experiment with the variables in the brightness and flicker functions to achieve a more desirable effect.

> "We need to free up the pins so that they can be used again, by calling pwm.stop and GPIO.cleanup"

The Code CANDLELIGHT EFFECT

```
#!/usr/bin/env python

# Import the modules used in the script
import random, time
import RPi.GPIO as GPIO

# Assign the hardware PWM pin and name it
led = 18

# Configure the GPIO to BCM and set it to output mode
GPIO.setmode(GPIO.BCM)
GPIO.setup(led, GPIO.OUT)

# Set PWM and create some constants we'll be using
pwm = GPIO.PWM(led, 100)
RUNNING = True
WIND = 9

def brightness():
    """Function to randomly set the brightness of
    the LED between 5 per cent and 100 per cent power"""
     return random.randint(5, 100)

def flicker():
    """Function to randomly set the regularity of the'flicker effect'"""
     return random.random() / WIND

print "Candle Light. Press CTRL + C to quit"

# The main program loop follows.
# Use 'try', 'except' and 'finally' to ensure the program
# quits cleanly when CTRL+C is pressed to stop it.
```

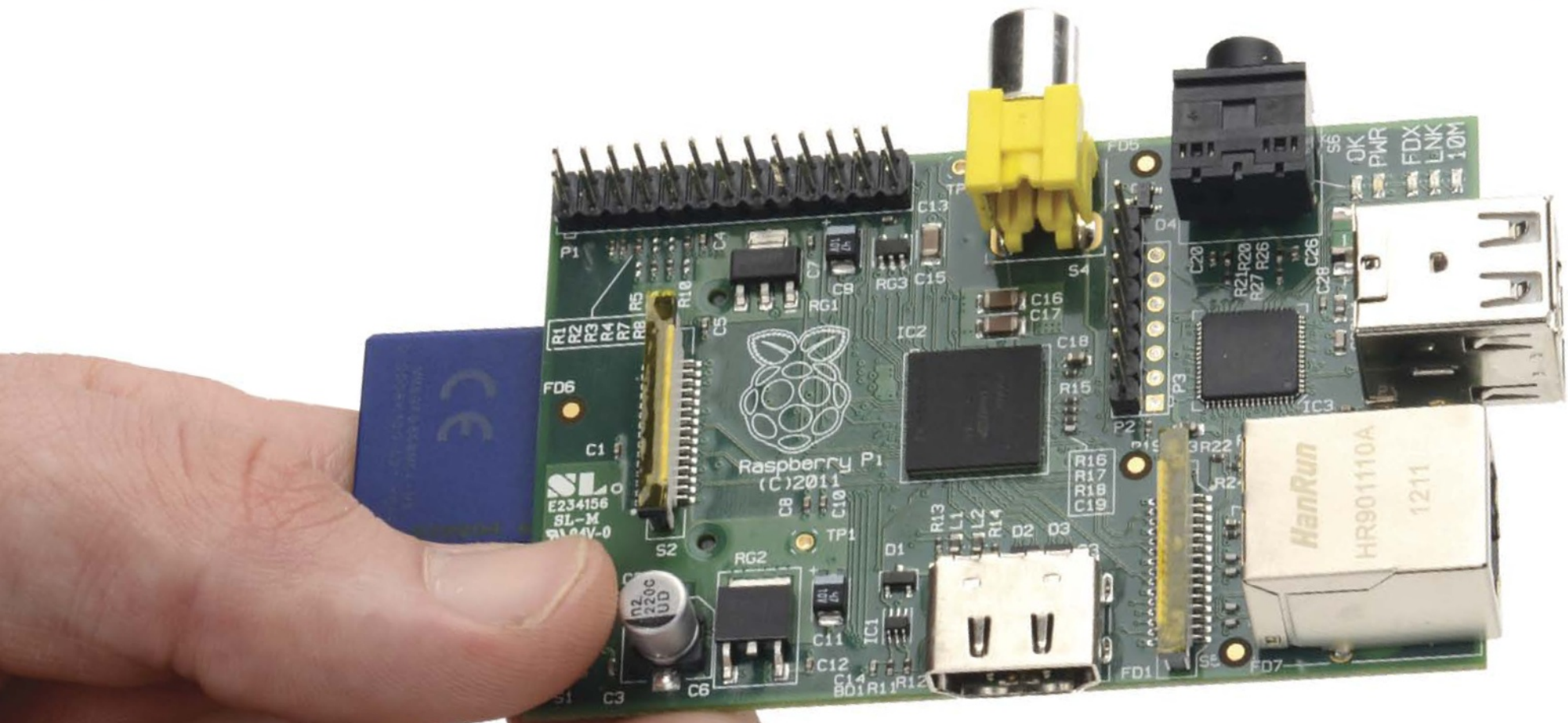
The Code CANDLELIGHT EFFECT

```
try:
    while RUNNING:
        # Start PWM with the LED off
        pwm.start(0)
        # Randomly change the brightness of the LED
        pwm.ChangeDutyCycle(brightness())
        # Randomly pause on a brightness to simulate flickering
        time.sleep(flicker())

# If CTRL+C is pressed the main loop is broken
except KeyboardInterrupt:
    running = False
    print "\nQuitting Candle Light"

# Actions under 'finally' will always be called, regardless of
# what stopped the program (be it an error or an interrupt)
finally:
    # Stop and cleanup to finish cleanly so the pins
    # are available to be used again
    pwm.stop()
    GPIO.cleanup()
```

Talking Pi

Join the conversation at...

@linuxusermag | Linux User & Developer | @ RasPi@imagine-publishing.co.uk



The Raspberry Pi encourages a lot of hands-on work and this means it doesn't quite work like – or as easy as – your laser focus-tested smartphone interface, or even a normal computer. So we're answering your burning Raspberry Pi questions each issue – get in touch with us on Twitter, Facebook or by email.

Will all the current software and projects for Raspberry Pi work with the new Raspberry Pi Model B+?
Tom W.
via email

All the software will work just fine as the Model B+ is essentially the same core hardware as the original Raspberry Pi. This includes the distros and any code or files you've created on another Raspberry Pi.

As for projects, anything requiring specific connectors will work just fine. The GPIO arrangement is the same, only with a few extras tagged on the end, and the only missing port is the Video-out, which only had very specific purposes. Anything using a custom case might have an issue though due to the new layout of the USB ports and HDMI port.

Is there a way for me to use Raspbian from my Model B with the Model B+?
Ravi
via Facebook

One of the easiest methods is to clone your current SD card and then write it to a microSD card using the same dd method we would normally use. You can also use dd to clone the original card using a command that is something like:

```
$ sudo dd if=/dev/sda[x]/ of=~/PiBackup.img
```

...where x would be the location of your SD card mounted on the system.

Keep up with the latest Raspberry Pi news by following @LinuxUserMag on Twitter. Search for the hashtag #RasPiMag

LinuxUserMag: The @Raspberry_Pi's education mission is excellent and tremendously important for kids. However, have any adults learnt by using it?

LinuxUserMag: We've learnt a lot about physical computing since we started using it so we don't think it's beyond the realm of possibility.

Roffesoft: @LinuxUserMag @Raspberry_Pi Its got me back into C programming after a lapse of 10+ years, does that count ?

KentGeek: I learnt some basic electronics, a lot about physical computing and how to use Pygame with mine, and I'm 27

I have read that you should make sure to get enough power to your Raspberry Pi. What does this mean?
Luce via email

While you can power a Raspberry Pi from a USB port on a computer, it usually won't be enough to properly power the Pi. This will limit its abilities, especially with other components connected. When it comes to power adapters they can all greatly differ – they can commonly be about 800 mA or 1 A if they're any good. To properly power your Pi, it's recommended that you get a 1.5 A power supply, if possible.

Is there a way to make my Raspberry Pi portable – portable enough to stick in a bag?
David Hart via Facebook

A lot of modern portable batteries for charging phones or tablets are usually up to the job of powering a Raspberry Pi – although refer to our previous question about power supplies to get an idea of how powerful they'll need to be. In terms of merely powering it on then this will do the trick, but if you need a display to go with it then you can either set up a VNC server on an Android phone or simply use one of Adafruit's PiTFT touchscreens that slots onto the Raspberry Pi.

chorltonheelies: @LinuxUserMag @Raspberry_Pi everything I know about Linux, Python, electronics and cheap Chinese power supplies is thanks to Raspberry Pi

rubinsteindds: @LinuxUserMag @Raspberry_Pi I just got an arduino kit and a raspberry pi, looking forward to playing with both, with my kids.

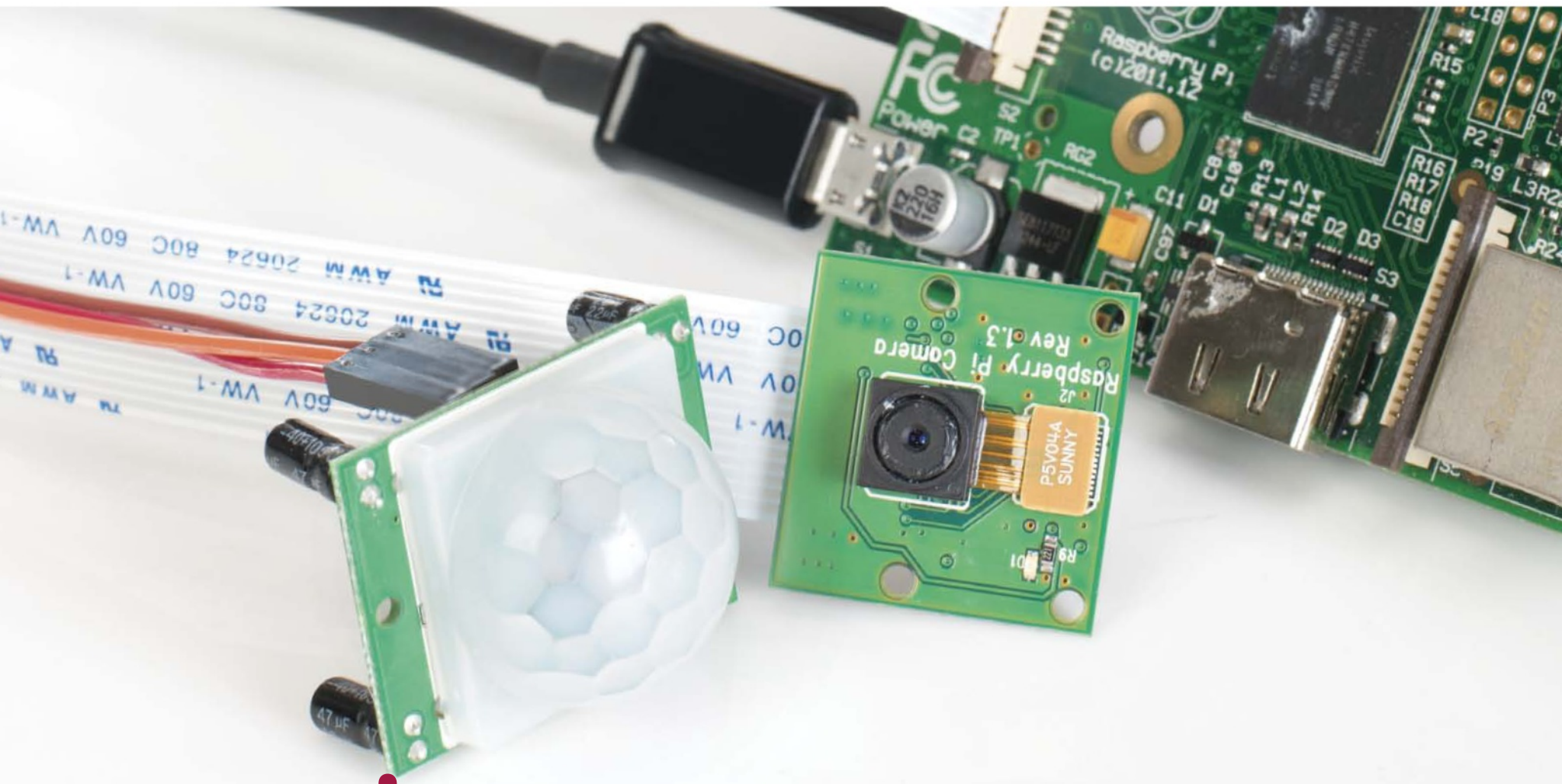
PiTutorials: @LinuxUserMag @Raspberry_Pi I've been trying to learn programming, failed many times. Now building a robot and learning python.

PiTutorials: @KentGeek @Raspberry_Pi @LinuxUserMag About the same here, a bit older. More adults using it= more kids will come in contact with it all!

I_am_Grand: @LinuxUserMag Yes! Networking in Python, OpenFrameworks in C++ & introduced me to Linux. Overambitious projects helped my learning :)

Next issue

Get inspired · Expert advice · Easy-to-follow guides



Make a tweeting motion sensor

Plus RasPi robot part 2 (sensors & inputs), 3-colour lamp and more

Get this issue's source code at:

www.linuxuser.co.uk/raspicode

FOR THE GNU GENERATION

www.linuxuser.co.uk

FREE RESOURCE DOWNLOAD EVERY ISSUE •

LinuxUser & Developer

Available from all good newsagents & supermarkets today

ON SALE NOW:

» IBM and Linux » Why you need Python 3 » 4 Great competitions

BUY YOUR ISSUE TODAY

Print edition available at www.imagineshop.co.uk

Digital edition available at www.greatdigitalmags.com

Available on the following platforms

Print | Tablet | Phone | Desktop

facebook.com/LinuxUserUK twitter.com/LinuxUserMag